FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Journal - webový osobní deník
Student:	Matyáš Herman
Vedoucí:	Ing. Zdeněk Rybola, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Proveďte rozšíření aplikace Journal, která vznikla v rámci týmového projektu. Zejména rozšiřte funkcionalitu aplikace o podporu lokalizace, označování osob v událostech, hierarchické události a monitoring aplikace.
Postupujte podle následujících pokynů:
- popište stávající funkce aplikace
- analyzujte nové požadavky a jejich začlenění do aplikace
- popište stávající architekturu aplikace a navrhněte realizaci nových požadavků
- implementujte nové funkce a náležitě otestujte
- připravte aplikaci k produkčnímu nasazení ve verzi 1.0 a aktualizujte dokumentaci

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 5. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Journal – webový osobní deník

Matyáš Herman

Katedra softwarového inženýrství

Vedoucí práce: Ing. Zdeněk Rybola, Ph.D.

15. května 2019

Poděkování

Děkuji svému vedoucímu Ing. Zdeňku Rybolovi, Ph.D., za ochotu vést moji bakalářskou práci a za cenné rady. Další poděkování patří všem, kteří se podíleli na vývoji prvních verzí aplikace Journal, na které jsem zde mohl navázat. V neposlední řadě bych chtěl poděkovat rodině a přátelům, kteří mne po celou dobu studia podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2019 ……

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

Herman, Matyáš. *Journal – webový osobní deník*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Tato práce se zabývá rozšířením aplikace *Journal*, která již vznikala po dobu dvou semestrů v předmětech BI-SP1, BI-SP2. Tato aplikace slouží k uchovávání událostí, s možností zobrazení událostí na mapě či v kalendáři. Cílem této práce je provést analýzu aktuálního stavu, návrh nových funkcionalit rozšíření a jejich následná implementace. Jedno z těchto rozšíření je lokalizace aplikace do českého a anglického jazyka. Mezi další rozšíření patří možnost přidávat podudálosti či označovat u událostí osoby. Výstupem práce je funkční verze, schopná provozu na serveru.

Klíčová slova Journal, webová aplikace, zaznamenání událostí, událost, deník, PHP, Nette, lokalizace

Abstract

This thesis deals with the extension of the *Journal* application, which has already been developed for two semesters in subjects BI-SP1, BI-SP2. Use of this application is to store events, with the ability to view events on a map or in a calendar. Work is aimed to analyze the current state, design new functionalities of extensions and their subsequent implementation. One of these extensions is the localization of the application to the Czech and English language. Other extensions include the ability to add sub-events or tag people at events. The output of the work is a functional version, capable of running on the server.

Keywords Journal, web application, recording events, event, diary, PHP, Nette, localization

Obsah

Seznam obrázků

Seznam tabulek

Seznam kódů

Úvod

Webové aplikace se stále vyvíjejí a jejich využívání den ode dne roste. V dnešní době, kdy mobilní telefon má již skoro každý a díky tomu má neustálý přístup na internet, jsou webové aplikace k dispozici kdykoli a odkudkoli.

Zaměřením mé bakalářské práce je aplikace *Journal*, která již vznikala po dobu dvou semestrů pod vedením Ing. Zdeňka Ryboly, Ph.D., a to v předmětech BI-SP1 (Softwarový týmový projekt 1) a BI-SP2 (Softwarový týmový projekt 2) na Fakultě informačních technologií. Tato aplikace slouží k online uchovávání a správě událostí, jako tomu bylo kdysi v deníčku. Zde se projevuje výhoda webové aplikace oproti papírovým deníkům, které uživatel musel nosit s sebou, nebo si zapisovat události později. *Journal* aplikace také umožňuje oproti papírové verzi deníku ukládat k událostem tagy a lokaci s následným zobrazením událostí na mapě, zobrazení událostí v kalendářním náhledu či vyhledávání jednotlivých událostí podle data/názvu/tagu.

První kapitola 1 této práce se zabývá seznámením s aktuální verzí aplikace, analýzou současného stavu a analýzou rozšířené aplikace. Po této části následuje kapitola 2 návrhová část, kde je představeno nové řešení aplikace a nových požadavků. Poslední kapitola je kapitola 3 realizace, která se zabývá vývojem samotné aplikace do finální podoby. Součástí této kapitoly jsou i příručky a následné testování aplikace.

Cílem této práce je rozšíření a zdokonalení aplikace, aby byla použitelná pro veřejnost. Tato bakalářská práce se tedy zaměřuje na přidání nových funkcionalit. Nová verze aplikace umožní uživatelům u událostí označovat osoby, a také si díky lokalizaci mohou zobrazit aplikaci jak v českém, tak i v anglickém jazyce. Další funkcí je rozlišování administrátorského účtu, který bude mít rozšířené možnosti, jako monitorování uživatelů, blokování a změnu práv

uživatelů, či rozesílání emailů ostatním uživatelům. V neposlední řadě bude aplikace rozšířena o hierarchické přidávání událostí. To znamená, že si uživatel může k události přidávat jiné podudálosti. Výstupem práce budou také elektronické příručky (uživatelská/vývojářská/nasazení) a Unit testy pokrývající nové funkcionality.

Kapitola 1

Analýza

V této kapitole bude provedena analýza stávajícího systému. Výsledek této analýzy poslouží jako výchozí bod pro další část této kapitoly, ve které bude popsána analýza nových funkcí aplikace. Tato kapitola zároveň pomůže k lepšímu pochopení a představení aktuální verze aplikace, v jakém stavu se nachází, co umí a k čemu ji lze využívat. Kapitola je rozdělena na čtyři části.

První část této kapitoly 1.1 se nazývá „Popis stávající aplikace“, a jak z názvu vyplývá, popisuje aktuální verzi aplikace *Journal*. Dále jsou zde popsány aktuální funkce aplikace 1.1.2 a doménový model 1.1.4. V druhé části 1.2 jsou popsány technologie, které byly použity při vývoji aplikace. Třetí část 1.3 se zabývá analýzou nových požadavků, které byly zadány vedoucím bakalářské práce a co vše bude potřeba udělat. Ve čtvrté, poslední části 1.4, je popsána konkurence pro naši aplikaci.

1.1 Popis stávající aplikace

Tato sekce popisuje, co aplikace *Journal* je a co vše uživatelům umožňuje. Tato část kapitoly se zabývá představením aplikace, tím jak vypadá po roce vývoje a jaké týmy za prací na aplikaci stojí.

1.1.1 Úvod a historie aplikace

Aplikace Journal se zrodila v předmětu BI-SP1 v roce 2018, kde ho založil Ing. Zdeněk Rybola, Ph.D. Tým, který se podílel na vývoji, byl ve složení Vojtěch Kraus, Matyáš Herman, Martin Vatrt a Tomáš Voldřich. Během tohoto předmětu vznikla analytická dokumentace a první prototyp aplikace.

Poté na jejich práci navázaly dva týmy v rámci předmětu BI-SP2. První tým pokračoval ve vývoji webové aplikace a to ve složení Vojtěch Kraus a Matyáš Herman. Práce probíhala jak na frontendu tak na backendu, a také bylo

vytvořeno *REST API* (Representational State Transfer Application Programming Interface) pro mobilní aplikaci. Postupně splňovaly další funkční požadavky, které byly vytvořeny během předmětu BI-SP1.

Druhý tým začal s prací na mobilní aplikaci pro operační systém Android, ve složení Igor Tsaregorodtsev a Zhanybek Sadvakassov. Nejedná se o aplikaci obsahující všechny funkce, jako má webová aplikace, ale umožňuje uživatelům mít hlavní funkce aplikace vždy k dispozici.

Během těchto dvou semestrů začaly vznikat první verze aplikace, které postupně splňovaly již stanovené funkční požadavky. Některé požadavky ale nefungují tak, jak bylo plánováno, a některé funkce ani nebyly implementovány.

1.1.2 Aktuální funkce aplikace

Tato sekce se zabývá aktuálními funkcemi webové aplikace. Budou zde představeny hlavní funkcionality aplikace.

1.1.2.1 Uživatelé

Aplikace poskytuje uživatelský profil. To zahrnuje registraci uživatele s aktivací účtu pomocí emailu, na který je poslán vygenerovaný token. Aplikace je použitelná pouze pro registrované a přihlášené uživatele. Po přihlášení má uživatel k dispozici všechny funkce aplikace, které budou popsány níže.

1.1.2.2 Události

Základem celé aplikace jsou události. Každý uživatel si po přihlášení může ukládat události. U událostí jsou implementovány základní *CRUD* (Create, Read, Update, Delete) operace. Ke každé události si může uživatel přidávat:

název – který slouží k identifikaci události

datum – určující datum od kdy, do kdy událost probíhala

tagy – umožňují seskupovat různé události do skupin. Může se například jednat o označení s jakými lidmi událost probíhala (např rodina/ kamarádi/kolegové atd.). Především poslouží k lepší organizaci, protože lze pomocí nich vyhledávat události.

popisek – slouží k zaznamenání informací. Uživatel si zde zapíše, co se danou událost stalo / co dělal a jiné užitečné informace. Je zde použit textový editor využívající technologie WYSIWYG (What you see is what you get), to znamená, že co napíšeme do editoru, se ve stejné podobě zobrazí po uložení. Editor podporuje nadpisy 5 úrovní, hypertextové odkazy, vkládání videa a fotografií rovnou do textového pole, tabulky a mnoho dalšího.

přílohu – přílohy si může uživatel přiložit ke každé události. Při editaci události pak může přílohy měnit. Jako přílohu lze přidat obrázky typu PNG (Portable Network Graphics), JPEG (Joint Photographic Experts Group) a nebo textové soubory, například typu PDF (Portable Document Format). Přílohy lze mazat v detailu události.

lokaci – lokaci lze přidat pomocí vyhledání podle jména místa, kde je i možnost vyhledávat v okolí do 10 km kolem umístěného markeru, nebo na mapě vybrat místo pomocí markeru. Toto místo bude uloženo jako souřadnice, ke kterému si uživatel může přidat název místa podle své volby.

1.1.2.3 Mapa

Události které mají uloženou lokaci si může uživatel zobrazit na mapě, kde se zobrazují markery s polohou, kde se jednotlivé události děly. Jestliže je na jednom místě více markerů, pak se shlukují a zobrazí se jen modré kolečko s počtem událostí, které jsou zde zahrnuty. Po kliknutí na číslo se mapa přiblíží na danou lokaci a zobrazí jednotlivé markery. Po kliknutí na marker se zobrazí dialogové okno s názvem události, popiskem a názvem lokace. Z tohoto dialogového okna lze přejít rovnou na detail události.

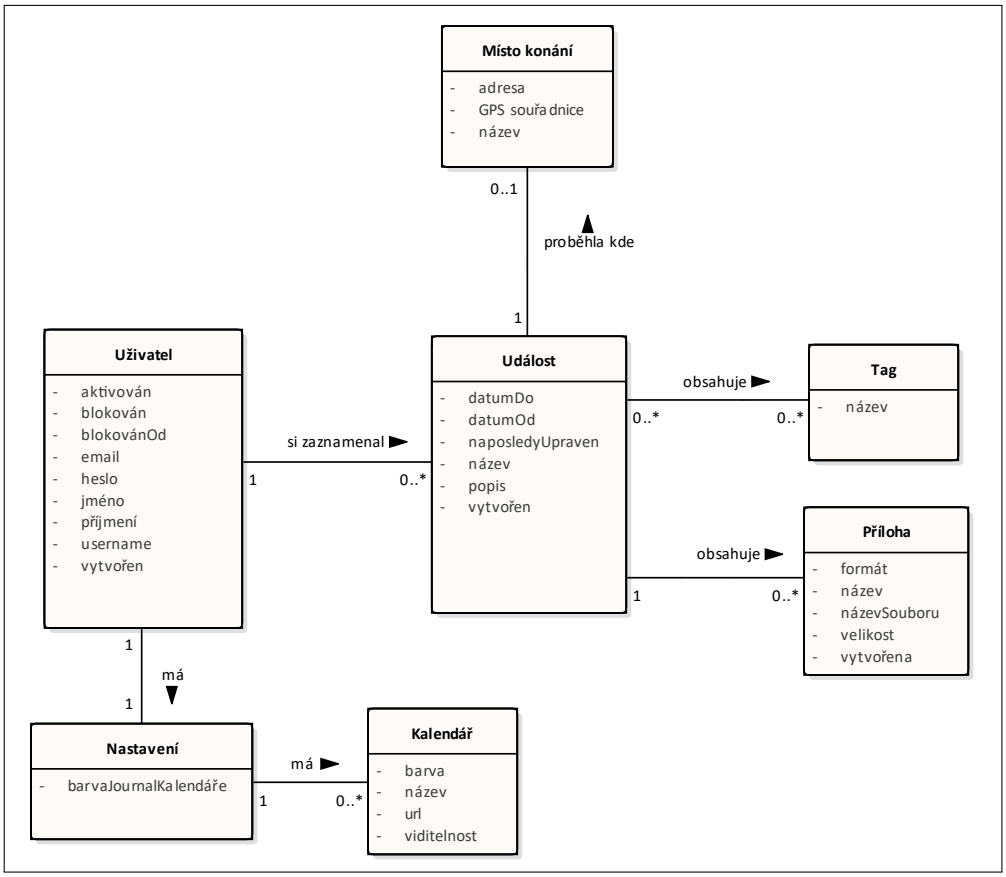
1.1.2.4 Kalendář

Aplikace má implementovaný kalendář, který zobrazuje všechny události, které si uživatel přidal v rámci aplikace. Kalendář podporuje měsíční zobrazení událostí. Každá událost z aplikace je zde zobrazena zelenou barvou přes dny kdy se odehrávala, a je zde napsán název události. Po kliknutí na událost se zobrazí modálové okno s jejím popisem. Je zde zobrazeno datum, název a popis události. Z modálového okna se lze prokliknout na detail dané události.

Kromě událostí z aplikace lze zobrazovat události z externích kalendářů. Externí kalendáře lze přidávat v nastavení, pomocí veřejné .ics *URL* (Uniform Resource Locator) kalendáře. Tyto externí kalendáře se poté zobrazí (barvou, kterou si uživatel určí v nastavení) v kalendáři aplikace. U každého externího kalendáře lze také v nastavení nastavit, zda se bude zobrazovat v kalendáři. V kalendáři po kliknutí na událost z externího kalendáře se zobrazí název, datum a popisek události v modálovém okně. Odtud se lze prokliknout na přidání nové události s již předvyplněnými informacemi (název, datum, popis) z externí události.

1.1.3 Mobilní aplikace

Mobilní aplikace byla vyvíjena pro operační systém Android. Aplikace zatím není ke stažení oficiálně z *Google Play*. Aplikace se zatím nachází pouze v soukromém projektu na *GitLabu* fakulty informačních technologií. Hlavní



Obrázek 1.1: Doménový model aplikace

funkce aplikace jsou zobrazení, přidání a úprava událostí. Je zde implementována možnost zobrazení událostí v kalendáři, včetně událostí z externích kalendářů. Mobilní aplikace není zcela dokončena, a v budoucnu se počítá s jejím dalším vývojem.

1.1.4 Doménový model

Další důležitou částí analýzy je doménový model. Tento model se často znázorňuje pomocí *UML* (Unified Modeling Language) diagramu tříd [1]. Neobsahuje žádné detaily ohledně implementace, ale poslouží jako dobrý výchozí bod pro modelování databázového modelu na obrázku 2.2 a návrhového modelu tříd 2.3. Model slouží pro lepší pochopení entit, které se nalézají v dané doméně, popisu jejich atributů a hlavně vztahů, které mezi jednotlivými entitami existují. Diagram na obrázku 1.1 zachycuje třídy, které souvisejí s analyzovanou doménou.

1.1.4.1 Uživatel

Tato třída reprezentuje uživatele zaregistrovaného v systému. Poslední aktualizace byla provedena v předmětu BI-SP2, kdy byly přidány nové atributy (Aktivován, Blokován, Blokován od, Vytvořen), potřebné pro správu uživatelských účtů. Entita *Uživatel* má vazbu s entitou *Nastavení* a s entitou *Událost*.

1.1.4.2 Událost

Každá událost je složena z názvu, popisku, data, tagů a dalších metadat a má jen a pouze jednoho majitele (uživatele), který může tuto událost upravit či odstranit. Entita *Událost* má tři vazby a to vazby s entitami *Místo konání*, *Tag* a *Příloha*.

1.1.4.3 Místo konání

Místo konání slouží k přidání lokace k události. Uchovává informace o názvu místa, adrese a GPS (Global Positioning System) souřadnicích lokace události.

1.1.4.4 Tag

Tag se přidává k událostem. Slouží k rychlé identifikaci a sdružení událostí se stejným tagem. Tag zastává vlastnost klíčových slov.

1.1.4.5 Příloha

Příloha reprezentuje dodatečné soubory, které se vztahují k prožité události. Jedná se například o fotografie, videa z výletů, či různé dokumenty, které byly pořízeny například během zaznamenané schůzky.

1.1.4.6 Nastavení

Nastavení je entita, starající se o barvu *Journal* kalendáře, která je zatím neměnná. Tato entita má vazbu s entitou *Kalendář*.

1.1.4.7 Kalendář

Entita Kalendář má na starosti informace o externích kalendářích. Tato entita byla přidána v předmětu BI-SP2. Pro každý kalendář si uchovává název, url, barvu a zda je viditelný.

1.2 Aktuální implementace aplikace

Až doposud se zde psalo o aplikaci pouze obecně, o vývoji, co umí atd. V této sekci je popsána architektura aplikace.

1.2.1 Backend

Celá backendová část aplikace, která se stará o práci s daty, je naprogramovaná v jazyce *PHP* [2] (Hypertext Preprocessor) společně s českým frameworkem *Nette* [3]. Třídy jsou do *MySQL* [4] (Structured Query Language) databáze mapované pomocí knihovny *Doctrine 2* [5], která je zároveň s dalšími závislostmi nainstalovaná pomocí *Composer* [6].

Backendová část aplikace je postavena na třívrstvé relaxované architektuře – viz obrázek 1.2 s diagramem. Třívrstvá architektura se dělí na:

prezentační vrstvu – prezenční vrstva se stará o zobrazení informací uživateli, často pomocí *HTML* (Hypertext Markup Language) stránky, které jsou doprovázeny *CSS* (Cascading Style Sheets) styly. Tato vrstva může kontrolovat vstupy od uživatele a stará se o změny v business vrstvě či pohledu.

business vrstvu – business vrstva se stará o logiku aplikace a je nezávislá na prezentační a datové vrstvě. Tato vrstva se nestará o to, odkud data přišla a jak budou jako výstupní data formátovány a vypsány.

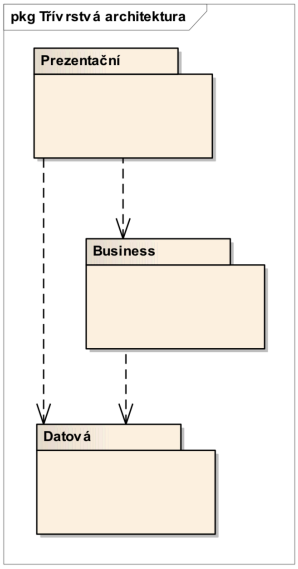
datovou vrstvu – datová vrstva je vrstva umožňující komunikaci mezi aplikací a datovým úložištěm.

1.2.2 Frontend

Frontend aplikace využívá šablonovacího systému *Latte* [7] pro *PHP*, který je zabezpečen před zranitelnostmi typu *XSS* (Cross-Site Scripting). *Latte* bylo vytvořeno speciálně pro framework *Nette*, stejně jako šablonovací jazyk *Twig* [8] pro *PHP* framework *Symfony* [9]. Vzhled webové stránky je doplněn styly za pomoci jazyka *CSS* a *Bootstrap* [10] balíčků. O dynamiku na stránkách se stará *JavaScript* s *jQuery* [11] knihovnou.

1.2.3 Model nasazení

Pro lepší pomoc s utvořením představy o aplikaci je zde přidán obrázek s modelem nasazení 1.3. Tento model slouží k zobrazení fyzické architektury aplikace a technologií potřebných k běhu aplikace. Kromě technologií, které již byly zmíněny je zde zapotřebí server *Apache* [12]. Při vývoji aplikace byla používaná distribuce *XAMPP* [13] (Cross-Platform, Apache, MariaDB,



Obrázek 1.2: Třívrstvá architektura [14]

PHP, Perl), která je pro operační systém *Windows* a distribuce *MAMP* [15] (Macintosh, Apache, MySQL, PHP), která je vyvinuta pro operační systém *macOS*.

1.3 Analýza nových požadavků

Nyní by čtenář měl být dostatečně obeznámen s funkcemi a fungováním aplikace *Journal*, a tak zde může být představeno co se bude během této bakalářské práce na aplikaci měnit.

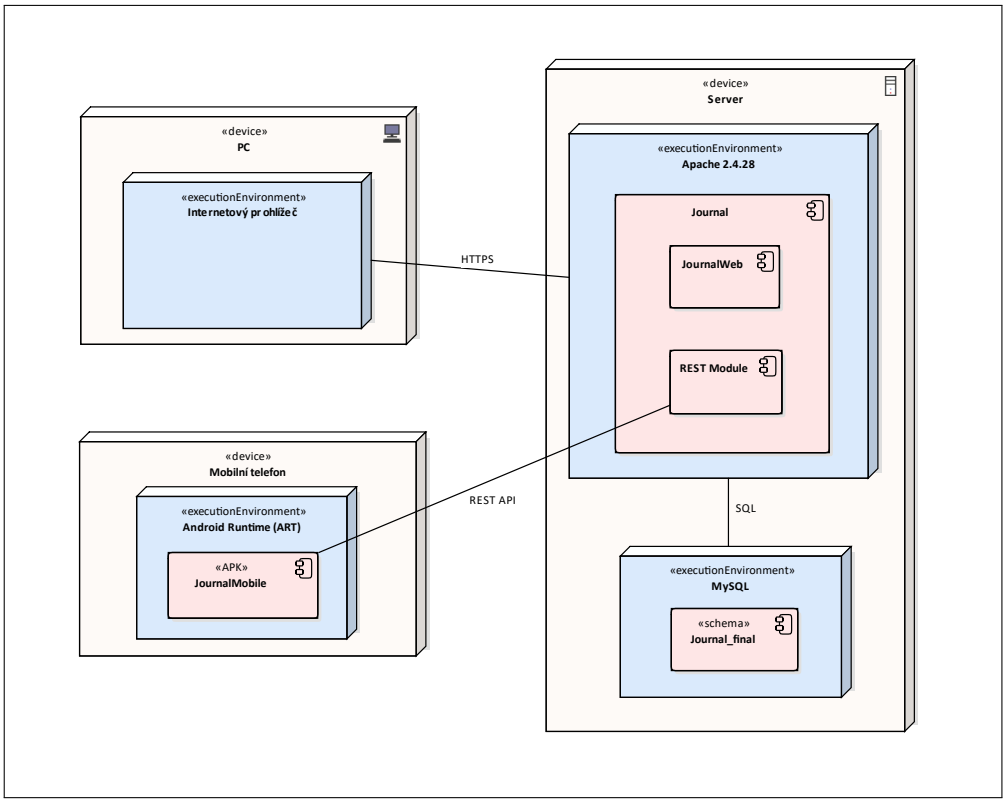
1.3.1 Specifikace nových požadavků

S vedoucím bakalářské práce byly již stěžejní nové funkční požadavky určeny v zadání práce. Poté bylo nalezeno několik dalších požadavků, které je potřeba doplnit či doladit.

1.3.1.1 Hlavní nové funkční požadavky

Zde jsou popsány nové funkční požadavky, značené FP (funkční požadavky), které byly určeny v zadání práce a jsou stěžejní pro tuto bakalářskou práci.

1. **FP1: Lokalizace**
    - Lokalizace aplikace znamená přizpůsobení aplikace pro různé jazyky. Aplikace bude rozšířena o podporu více jazyků. V této bakalářské práci bude přizpůsobena pro češtinu a angličtinu.



Obrázek 1.3: Model nasazení

- Přeloženy budou všechny stránky, včetně šablon emailů pro registraci a zapomenuté heslo. Tyto emaily se budou posílat v jazyce, ve kterém uživatel aktuálně prochází aplikaci.
- V sidebaru[1] aplikace bude možnost měnit jazyk z češtiny do angličtiny a obráceně.

2. **FP2: Označování osob**

- U událostí půjde označovat osoby. Osoby se budou ukládat a půjde je upravovat, či podle nich vyhledávat události.
- Ke každé osobě si může uživatel uložit název a email.
- Při vytváření události se již jednou přidané osoby budou nabízet k označení.

[1] postranní menu aplikace

- Bude vytvořena stránka se seznamem osob, na které si mohou uživatelé přidávat a upravovat osoby. Z tohoto seznamu se půjdou osoby i mazat.
- Ze seznamu se bude možné přes osoby prokliknout na seznam událostí, ve kterých je osoba označena.
- V detailu události budou vypsány všechny označené osoby. Po kliknutí na osobu bude uživatel přesměrován na seznam událostí, ve kterých je tato osoba označena.

3. **FP3: Monitoring aplikace**
   - Jelikož se aplikace bude spouštět na produkčním serveru, musí mít majitel aplikace přehled o uživatelích a chodu aplikace.
   - Bude vytvořena stránka se seznamem všech zaregistrovaných uživatelů.
   - Na stránce půjde vyhledávat podle jména, příjmení, mailu či pomocí stavu uživatele, zda je administrátor, blokovaný nebo aktivovaný.
   - Administrátor bude moci účty blokovat, aktivovat, přidat a odebrat práva administrátora či uživatelský účet úplně odstranit na žádost uživatele.
   - Také bude vytvořena stránka s přehledem chodu aplikace, na které administrátor uvidí aktivitu uživatelů v aplikaci. Bude zde přehled počtu nových uživatelů, událostí a dalších užitečných věcí, které lze poskytnout (například počet tagů, počet příloh a celková velikost příloh pro lepší kontrolu datového úložiště).
   - Administrátor nemá přístup k osobním informacím uživatelů, jako jsou události, osoby či přílohy.
4. **FP4: Hierarchické členění událostí**
   - Bude umožněno události hierarchicky uspořádat do dvou úrovní.
   - Hlavní (dále „rodičovská“) událost bude moci mít neomezené množství takzvaných podudálostí (dále „dětských“).
   - V seznamu událostí budou rozlišeny rodičovské a dětské události pomocí ikony u názvu události. Událost bez ikony bude normální jednoúrovňová událost.
   - Vytvořit dětskou událost půjde z detailu hlavní události i z jednoúrovňové události, která se pak stane rodičovskou událostí.

- V detailu rodičovské události bude seznam dětských událostí, které budou rozkliknutelné a přesměrují uživatele na dětskou událost.
- V detailu dětské události bude napsán název hlavní události, který bude prokliknutelný na detail rodičovské události.

1.3.1.2 Další funkční požadavky

Ze čtyř stěžejních funkčních požadavků vycházejí vedlejší funkční požadavky, které je potřeba splnit. Dále tedy na žádost vedoucího práce a z vlastní iniciativy budou provedeny následující změny, které navazují na předchozí sekci s funkčními požadavky. Proto nejsou číslovány od jedné.

5. **FP5: Administrátorský účet**
   - Pro možnost monitoringu je třeba zařídit administrátorský účet, aby neměl k informacím a správě uživatelů přístup každý.
   - Administrátorský účet bude úplně stejný jako normální účet, a navíc bude mít administrátorské vlastnosti.
   - Administrátorská práva může uživatel získat změnou atributu *administrátor* v databázi, nebo mu je může přidělit jiný administrátor.
6. **FP6: Ztracené heslo**
   - Při ztraceném heslu neměl uživatel možnost získat nové heslo, což je velký nedostatek aplikace.
   - Při žádosti o nové heslo musí uživatel vyplnit email, kterým se registroval. Na email mu poté přijde odkaz s uživatelovým hashem na formulář, kam uživatel musí zadat nové heslo.
7. **FP7: Úvodní obrazovka**
   - Při příchodu na stránku aplikace se pouze zobrazí přihlašovací formulář. Uživatel tedy vůbec neví, o jakou aplikaci se jedná a k čemu slouží. Je tedy potřeba vytvořit úvodní obrazovku, která uživatele seznámí s aplikací.
8. **FP8: Informace o aplikaci**
   - Bude vytvořena stránka s informacemi o aplikaci, jak vznikala a kdo za aplikací stojí.
   - Do postranního sloupce v aplikaci bude přidána informace o verzi aplikace a email na administrátora aplikace.
   - Bude přidána stránka s uživatelským manuálem. Pokud bude uživatel administrátor, bude mít navíc ještě příručku pro administrátora.

9. **FP9: Smluvní podmínky**

   - Jelikož bude aplikace brzy spuštěna na produkčním serveru, je z právního hlediska potřeba vědět, s čím uživatel souhlasil při registraci a jak je nakládáno s uživatelskými daty, které do aplikace zadá.
   - Smluvní podmínky budou uloženy v databázi a administrátor bude moci přidávat nové smluvní podmínky a nastavovat jim datum, od kterého budou platné.

10. **FP10: Vylepšení kalendáře**

    - Externí kalendáře jde vypínat/zapínat pouze v nastavení. To je pro uživatele velmi zdlouhavé.
    - Na stránce kalendáře budou vytvořeny checkboxy s názvy externích kalendářů. Těmito checkboxy se budou kalendáře vypínat a zapínat přímo na stránce kalendáře, což uživateli ušetří mnoho času.
    - V nastavení bude vypínání/zapínání kalendářů ponecháno. Toto nastavení bude výchozí pro zobrazení kalendáře.

11. **FP11: Přepracování detailu události**

    - Stránka detailu události je graficky nepovedená. A jelikož se v detailu události bude vypisovat mnoho nových informací, jako hlavní událost / podudálost, označené osoby, bude třeba přepracovat vzhled této stránky, aby odpovídal zbytku aplikace.

1.3.2 Nové případy užití

V této kapitole jsou nové případy užití, značené UC (use case), pro funkční požadavky popsány v sekci 1.3.1.1. Pro případy užití jsou popsány pouze hlavní scénáře. V případě výskytu chyby ve scénáři je uživateli zobrazena hláška.

1. **UC1: Změna jazyka:** změna jazyka umožňuje uživateli změnit jazyk aplikace za pomoci ikonek v sidebaru aplikace.

   Basic path: Změna jazyka – Uživatel změní jazyk aplikace.

   a) Uživatel klikne na jazyk, do kterého chce aplikaci přepnout.

   b) Systém zobrazí stránku na které byl uživatel v nově zvoleném jazyce.

2. **UC2: Označení osoby:** označení osoby umožňuje uživateli označit u události osobu. Osoby které již má uživatel uložené, se při psaní nabízejí.

   Basic path: Označení osoby – Uživatel do události označí osobu.

   a) Případ užití začíná, když se uživatel rozhodne přidat událost.

   b) Systém zobrazí formulář s atributy (datum, název, popis, tagy, přílohy, lokace a osoby)

   c) Uživatel vyplní informace o události a vyplní osoby, které chce u události označit. Při psaní osob se již uložené osoby v aplikaci nabízejí.

   d) Systém zkontroluje, zda jsou vyplněny povinné atributy, a zda jsou správně vyplněny a událost uloží. Když je v události přidána nová osoba, kterou uživatel v aplikaci ještě nemá, vytvoří se zcela nová osoba a uloží se. Přidá-li uživatel k události osobu, kterou již v aplikaci má, tak se k této osobě přidá záznam o nové události.

3. **UC3: Blokace uživatele:** blokace uživatele umožňuje administrátorovi zablokovat uživatele.

   Basic path: Blokace uživatele – Administrátor zablokuje uživatele.

   a) Případ užití začíná, když se administrátor rozhodne zablokovat uživatele. Klikne na stránku práva uživatelů.

   b) Systém zobrazí všechny uživatele a formulář pro vyhledávání uživatelů.

   c) Administrátor vyplní jméno a příjmení hledaného uživatele.

   d) Systém najde uživatele podle zadaných parametrů a zobrazí je.

   e) Administrátor najde uživatele a klikne na ikonku ve sloupci „blokován“.

   f) Systém uživatele zablokuje a odešle mu informační email o tom, že byl zablokován. Systém znovu zobrazí stránku, již se zablokovaným uživatelem.

4. **UC4: Vytvoření podudálosti:** vytvoření podudálosti umožňuje uživateli vytvořit podudálost nějaké události. Tyto podudálosti lze vytvářet již z rodičovské události nebo z události, která není potomkem jiné.

   Basic path: Vytvoření podudálosti – Uživatel vytvoří novou událost, která bude podudálostí jiné.

   a) Případ užití začíná, když uživatel zažije událost a rozhodne se ji uložit jako podudálost jiné události.

   b) Systém zobrazí seznam událostí.

   c) Uživatel vybere událost, která bude hlavní událostí pro jeho podudálost.

   d) Systém zobrazí detail události.

   e) Uživatel zvolí přidat podudálost. Možnost zvolit podudálost i z hlavního seznamu událostí je v řádku s událostí.

   f) Systém zobrazí formulář pro přidání události, navíc si uloží, že jde o podudálost určité události.

   g) Uživatel vyplní informace o události a uloží ji.

   h) Systém zkontroluje, zda jsou vyplněny povinné atributy, a zda jsou vyplněny všechny atributy správně. Poté událost uloží jako podudálost. A přesměruje uživatele na její detail.

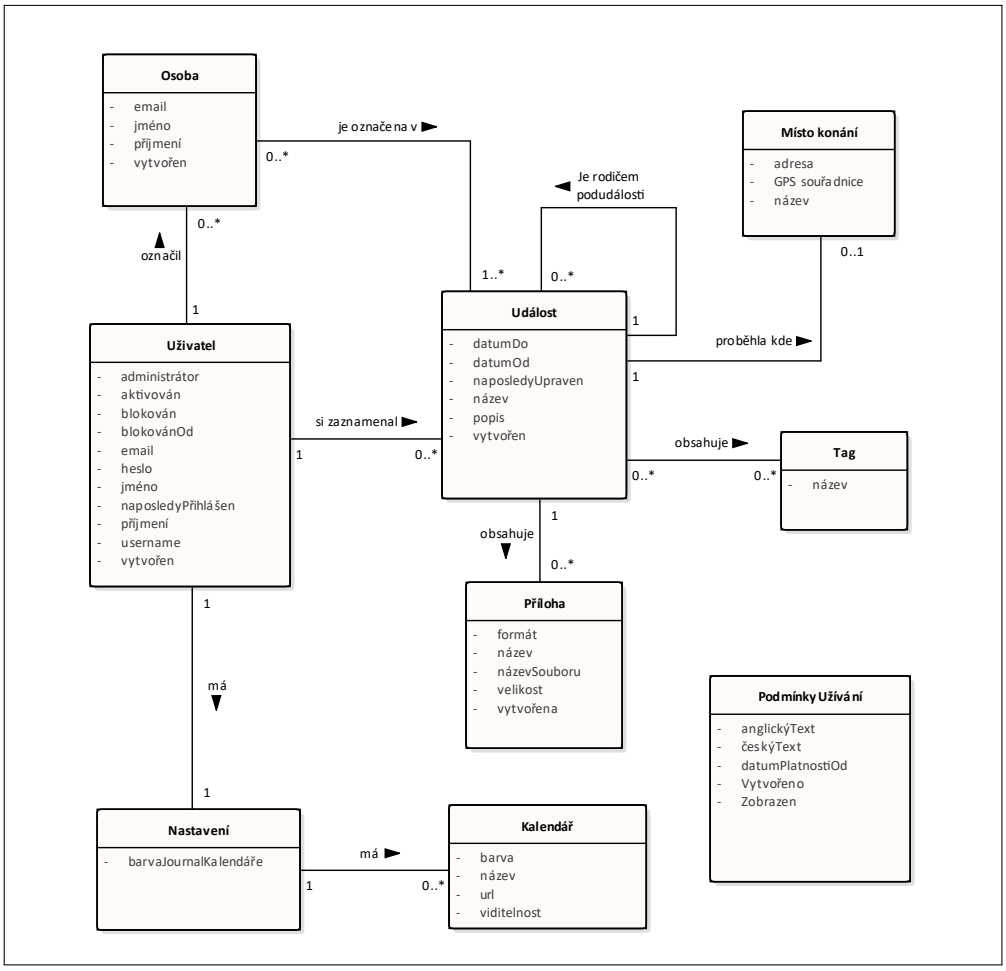
1.3.3 Rozšířený doménový model

Co je doménový model již bylo řečeno. Ukázán byl doménový model výchozího stavu aplikace na obrázku 1.1. Nyní je zde představena rozšířená verze doménového modelu na obrázku s diagramem 1.4. Přibyly zde nové entity s novými atributy a nové vazby. V této sekci jsou popsané jen nové a změněné entity.

1.3.3.1 Uživatel

Třída *Uživatel* byla pouze rozšířena o nové atributy, které jsou velmi důležité pro rozšíření aplikace. Tyto atributy pomohou při vývoji administrátorských účtů. Jedná se o atributy:

administrátor – atribut *administrátor* je atribut, který rozhoduje o tom, zda je uživatelský účet administrátorský.



Obrázek 1.4: Rozšířený doménový model aplikace

naposledy přihlášen – atribut *naposledy přihlášen* je povinný. Poslouží k monitorování aktivity, aby administrátor mohl vidět, jaké účty jsou aktivní. Pokaždé, když se uživatel přihlásí, tak se zaznamená nové datum přihlášení.

Dále byla přidána vazba mezi touto entitou a entitou *Osoba*.

1.3.3.2 Událost

V entitě *Událost* nebyl přidán ani odebrán žádný atribut. Změna, která se týká této tabulky, je v přidání nové rekurzivní vazby pro vyjádření podudálostí, což je využito při vytváření hierarchických událostí. Dále byla přidána vazba na entitu *Osoba*, pro možnost přidávat osoby k událostem.

1.3.3.3 Osoba

Jedna z nových entit, které byly přidány do doménového modelu, je entita *Osoba*, která zastupuje osoby přidávané k událostem. Každý uživatel může mít několik osob a ty jsou pak označovány v událostech. V této entitě jsou pouze tři atributy:

jméno – atribut *jméno* je povinný atribut sloužící k zaznamenání názvu dané osoby

email – atribut *email* je atribut pro zaznamenání emailu osoby, je nepovinný

vytvořen – atribut *vytvořen* je povinné datum vytvoření osoby

1.3.3.4 Podmínky užití

Druhá a poslední entita, která byla nově přidána do doménového modelu je entita *Podmínky užití*. Tato entita poslouží ke splnění funkčního požadavku FP9 – Smluvní podmínky. Tato entita zastupuje jednotlivé smluvní podmínky, z nichž vždy právě jedna bude aktivní (zobrazena v aplikaci jako smluvní podmínky). Tato entita má atributy:

českýText – atribut *českýText* je povinný atribut sloužící k zaznamenání smluvních podmínek v českém jazyce.

anglickýText – povinný atribut *anglickýText* je atribut sloužící jako atribut *českýText* k zaznamenání smluvních podmínek, ale v anglickém jazyce.

datumPlatnosti – atribut *datumPlatnosti* slouží k zaznamenání data, od kterého jsou podmínky platné. Jedná se o povinný atribut.

zobrazen – povinný atribut *zobrazen* slouží k rozeznání aktuálně platných podmínek, které se v aplikaci zobrazují od ostatních nezveřejněných podmínek.

vytvořen – atribut *vytvořen* je povinné datum vytvoření nových podmínek.

1.4 Analýza konkurence

Jelikož se nejedná o přelomovou technologii ani o nový nápad, tak již několik aplikací používaných pro podobné či stejné účely existují. V této sekci budou představeny některé konkurenční aplikace včetně těch, ze kterých tato aplikace vychází. Informace o konkurenčních aplikacích byly získány přímo z webových stránek aplikací a z porovnání některých aplikací v článku z webové stránky Lifewire [16].

1. **Journey** – `https://journey.cloud`
   Aplikace *Journal* vychází převážně z této aplikace. Tato aplikace patří k těm nejrozšířenějším a je vyvíjena jako webová aplikace, desktopová aplikace a aplikace pro mobilní operační systémy *iOS* a *Android*. Aplikace za poslední rok vytvořila prémiovou verzi aplikace, ta vznikla tak, že většina funkcí z původní verze byla přesunuta do placené verze. Bezplatná verze tedy disponuje pouze přidáváním událostí s názvem, popiskem, tagy. Události si potom lze zobrazit pouze na časové ose.

   V prémiové verzi, která stojí $2,49 měsíčně, je navíc možno přidat k události polohu, počasí, videa, zvukový záznam. Události si pak lze navíc zobrazovat v kalendáři a na mapě. Jelikož všechny původní funkce aplikace jsou nyní placené, tak byl vytvořen prostor pro vytvoření aplikace, která bude mít tyto funkce a bude zcela bezplatná. To je důvod, proč vznikla aplikace *Journal*, na kterou navazuje tato bakalářská práce. [17]

2. **Diarium** – `https://timopartl.com`
   Aplikace *Diarium* je nejlépe hodnocená aplikace na uchovávání vzpomínek ve *Windows 10 App Store*. Lze zde také přidávat události s polohou, fotkami, soubory, tagy a navíc lze nahrávat zvukové záznamy. Události si pak lze zobrazovat i v kalendáři či na mapě. Události z aplikace lze exportovat[2] do DOCX, HTML či textového souboru.

   Tato aplikace je vyvinuta pouze jako *Windows* aplikace a aplikace pro operační systém *Android*. Proto je nevyhovující pro větší použití a nelze ji používat jako webovou aplikaci. Aplikace je zcela bez reklam. [18]

3. **Penzu** – `https://penzu.com`
   Aplikace *Penzu* je považována za nejbezpečnější aplikaci pro uchovávání událostí, a na této informaci si hodně zakládá. Tato aplikace je vyvinuta jen jako webová aplikace a aplikace pro mobilní operační systémy *Android* a *iOS*. Má přizpůsobitelný vzhled pozadí a textových fontů a má snadné vkládání obrázků přímo do textu. Při psaní události se text automaticky ukládá a počítá slova. K událostem lze přidávat fotografie a lokaci, ale tagy či videa přidávat nelze.

   Aplikace má také placenou verzi a to za $4,99 měsíčně nebo $19.99 za rok. V rozšířené verzi je podporovaný export událostí do PDF, obnovování událostí z koše či neustále možná podpora. U této aplikace si ale lidé stěžují, že často nefunguje a seká se při ukládání událostí. Události lze z aplikace, jak webové tak mobilní, sdílet s přáteli přes Facebook nebo email. Mezi velké nevýhody patří občas vyskakující reklamy v aplikaci. [19]

[2] Převedení souboru do jiného formátu, než v jakém je.

4. **Diary** – `https://www.writediary.com`
   Aplikace *Diary* je dostupná jako webová aplikace a aplikace pro mobilní operační systém *Android*. V aplikaci lze přidávat události pouze s názvem, datem a popiskem. Do popisku události i názvu lze vkládat emoji[3]. Aplikace je zcela bezplatná, ale oproti jiným nenabízí mnoho funkcí, které by aplikaci něco přidaly jako třeba přidávání fotografií, tagů či lokace k události nebo možnost si zobrazit události v kalendářním anebo mapovém zobrazení.

   Vývojáři aplikace se spíše zaměřují na vývoj mobilní aplikace, kde si lze i měnit barvu pozadí a další. V mobilní aplikaci je také nastavitelné upozornění, aby uživatel nezapomněl napsat událost v daný den. [20]

5. **Day One** – `https://dayoneapp.com`
   Aplikace *Day One* je vytvořena pouze pro *Apple* produkty, je tedy ke stažení na mobilní operační systém *iOS* v *App Store* a pro operační systém *macOS* v *Mac App Store*. Služba neposkytuje webovou aplikaci, což jí velmi ubírá na použitelnosti. V této aplikaci lze vytvářet události s pouze jednou fotografií a nelze přidávat videa, lokaci ani tagy. Události lze vyexportovat do různých formátů – PDF, JSON, čistý text.

   Aplikace má i placenou verzi, která má navíc tmavý režim, přidávání zvukových stop k událostem, kreslení pomocí Apple pencil, a podporu více fotografií u události. Cena placené verze je $2,92 měsíčně. [21]

6. **Five Minute Journal** – https://www.intelligentchange.com/
   Five Minute Journal je aplikace pouze pro mobilní operační systém *Android* a *iOS*. Tato aplikace slouží k zaznamenávání událostí, ale více se zaměřuje na plánování budoucích událostí. K událostem lze přidávat pouze fotografie. V aplikaci je možné nastavit upozornění na ráno a večer, aby si uživatel naplánoval svůj den. Vše lze z aplikace vyexportovat do PDF formátu. Aplikace je placená jednorázovým poplatkem $4,99. [22]

7. **Diaro** – `https://diaroapp.com`
   Diaro je celkem rozšířená aplikace podporující webové rozhraní a operační systémy *iOS* a *Android*. U událostí lze přidávat klíčová slova, neomezený počet fotografií a lokaci. Události lze řadit do složek, pomocí kterých pak lze vyhledávat a lépe události strukturovat. Vyhledávat v událostech také lze za pomoci data, lokace (a již zmíněných složek). Události lze jednoduše sdílet i lidem, kteří aplikaci nepoužívají a nemají v ní účet.

[3] Symbol vyjadřující aktuální náladu či pocity

> V aplikaci se nacházejí reklamy, ale je zde i placená verze, ve které nejsou reklamy a platí se za ni jednorázový poplatek $5,99. Placená verze pak přináší navíc exportování událostí do PDF, TXT, DOCX či synchronizaci pomocí cloudových služeb. [23]

V následujícím tabulkovém přehledu 1.1 se nachází porovnání výše zmíněných aplikací od konkurence. Aby byla tabulka více přehledná, budou zde především porovnány vlastnosti aplikací bez prémiových rozšíření. Funkce, které jsou v aplikaci jiné po zaplacení prémiové verze jsou označeny znakem „*“. V posledním řádku tabulky jsou subjektivně seřazeny aplikace podle toho, jak moc konkurují aplikaci *Journal* od 1 do 7 (1 - nejvíce, 7 - nejméně). Operační systém *Android* je v tabulce zkrácen na And a operační systém *Windows* na Win.

Výsledkem této analýzy je, že většina existujících aplikací nemá funkce, které má či bude mít aplikace *Journal*. Tato aplikace má oproti ostatním aplikacím možnost si importovat událost z externích kalendářů či si události zobrazit na mapě nebo v kalendáři a to vše v bezplatné verzi. Další výhodou oproti ostatním aplikacím je možnost si označovat v událostech osoby, pomocí kterých si pak lze události lépe organizovat. Výhodou této aplikace je český jazyk, kterým ostatní aplikace nedisponují nebo přidávání libovolných příloh (například fotografie, dokumenty).

	Journey	Diarium	Penzu	Diary	Day One	Five Minute	Diaro
Webová aplikace	ANO	NE	ANO	ANO	NE	NE	ANO
Mobilní aplikace	And, iOS	Win, And	And, iOS	And	iOS	And, iOS	And, iOS
Desktopová aplikace	ANO	ANO	NE	NE	ANO	NE	NE
Přidání fotografií	ANO	ANO	ANO	NE	pouze jedna*	ANO	neomezeně
Přidání tagů	ANO	ANO	NE	NE	ANO	NE	ANO
Přidání videa	NE*	NE	NE	NE	NE	NE	NE
Přidání audia	NE*	ANO	NE	NE	NE*	NE	NE
Přidání lokace	NE*	ANO	ANO	NE	ANO	NE	ANO
Vyhledávání	NE	ANO	ANO	ANO	ANO	NE	ANO
Reklamy v aplikaci	NE	NE	ANO*	NE	NE	NE	ANO*
Export událostí	NE	ANO	NE*	NE	NE*	ANO	NE*
Kalendářové zobrazení	NE*	NE	NE	NE	ANO	ANO	NE
Mapa	NE*	ANO	NE	NE	NE	NE	NE
Označování osob	NE	NE	NE	NE	NE	NE	NE
Český jazyk	NE	NE	NE	NE	NE	NE	NE
Placená verze	$2,49/měsíc	NE	$4,99/měsíc	NE	$2,92/měsíc	$4,99/jednou	$5,99
Seřazení	1.	3.	6.	4.	5.	7.	2.

Tabulka 1.1: Porovnání funkcí konkurenčních aplikací. Pozn: And znamená Android a Win znamená Windows.

Kapitola 2

Návrh

Po kapitole zabývající se analýzou aplikace je zde kapitola zabývající se návrhem nové verze aplikace. Za pomoci vhodných diagramů je zde představen finální návrh aplikace.

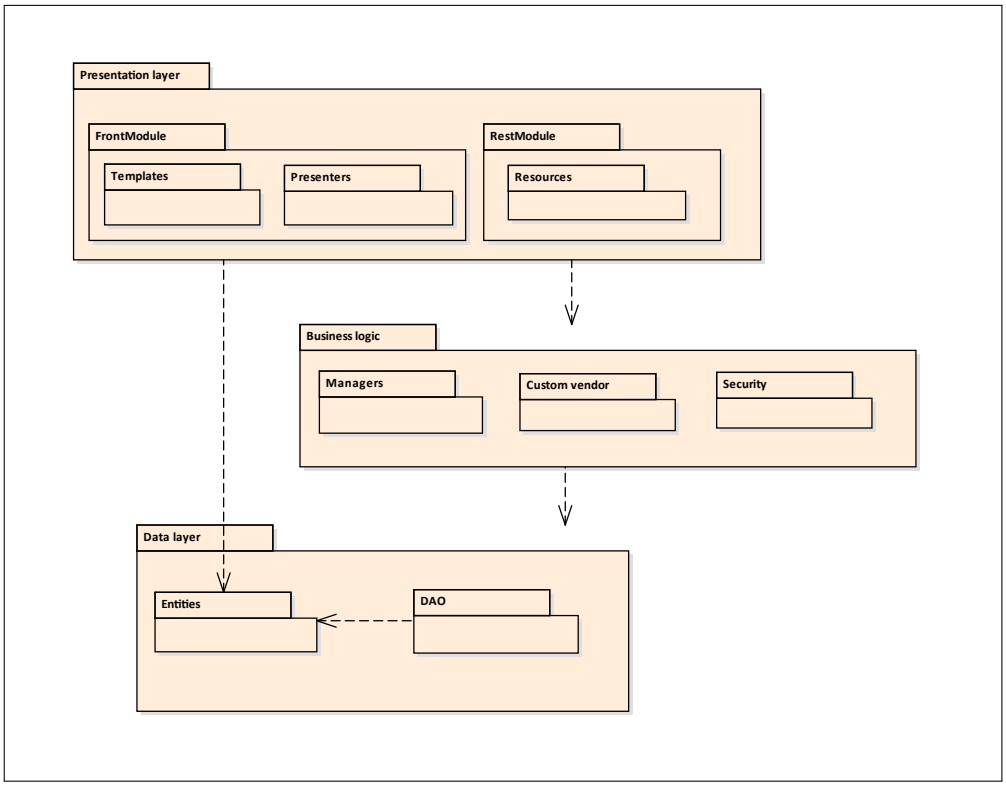
Nejprve je představena architektura aplikace a z jakých komponent se skládá. Poté je představen databázový model, který vychází z rozšířeného doménového modelu v sekci 1.3.1.1. V databázovém modelu jsou popsány rozdíly oproti původní verzi modelu. Pro pochopení záměrů je poté popsán třídní model, na kterém je demonstrován příklad na výsledné aplikaci. Na konci kapitoly je popsán ještě model komunikace, ve kterém je zobrazen sekvenční diagram pro tentýž příklad.

I přes to, že zamýšlené změny v aplikaci jsou docela rozsáhlé, nový návrh aplikace vychází z předchozích návrhů, které byly vytvořeny dříve při práci na tomto projektu. Proto jsou zde zobrazovány obrázky s diagramy související pouze se změnami v průběhu bakalářské práce, neboť navazují již na existující diagramy.

2.1 Architektura

Architektura je navržena jako třívrstvá MVC (Model-View-Controller) architektura, pro *Nette* framework se jedná o MVP (Model-View-Presenter), a byla již popsána v sekci 1.2. Komunikace mezi jednotlivými vrstvami je zajištěna pomocí rozhraní `interfaces`. Takto zvolená architektura umožní oddělený vývoj jednotlivých komponent bez nutné znalosti celé struktury aplikace. Je tím zároveň ulehčena výměna či rozšíření jednotlivých komponent.

V této sekci jsou popsány vrstvy aplikace. Pro lepší představení je zde přidán obrázek 2.1 s diagramem, který danou architekturu zobrazuje.



Obrázek 2.1: Návrh architektury

2.1.1 Prezentační vrstva

Prezentační vrstva zpracovává požadavky od uživatele a zobrazuje mu příslušný obsah. Pro zpracování uživatelských akcí komunikuje prezentační vrstva s logickou vrstvou pomocí rozhraní. Tyto požadavky jsou logické vrstvě předány bez ohledu na konkrétní návrh prezentační vrstvy, zda jde o webovou či mobilní aplikaci. Prezentační vrstva se skládá z následujících balíčků:

FrontModule – `FrontModule` je balíček, který obsahuje veškeré balíčky, starající se o prezentační vrstvu webové aplikace.

- **Presenters** – balíček `presenters`, obsahuje třídy, které budou zpracovávat dotazy uživatele. Třídy komunikují s logickou vrstvou pro získání odpovědí na uživatelův dotaz. Pomocí těchto informací vygeneruje příslušný `presenter` šablonu, kterou vrátí jako grafický výstup. Logické vrstvě budou zároveň z těchto tříd posílány veškeré informace o změnách provedených uživatelem.
- **Templates** – balíček `templates` neboli šablony, které využívají *Latte* [7] šablon, jsou použité k vygenerování příslušných statických stránek

zobrazených uživateli. Každá šablona obsahuje rozpis všech elementů, stylů a určuje jejich umístění na stránce.

RestModule – balíček `RestModule`, který se stará o prezentační vrstvu mobilní aplikace.

Resources – `resources` je balíček, ve kterém se nacházejí třídy, které implementují *REST API* aplikace, které mohou využít třetí strany pro vytváření a získávání objektů. Tento balíček obsahuje třídy, které obsluhují *HTTP* (Hypertext Transfer Protocol) requesty zasílané třetími stranami přes webové služby. Každá třída se stará o jeden `resource`, neboli objekt, se kterým chce volající nějakým způsobem manipulovat. Funkčně tyto třídy odpovídají třídám `presenters`.

2.1.2 Logická vrstva

Balíčky této vrstvy tvoří logiku celé aplikace. Jakékoliv složitější úkony, které je potřeba v aplikaci provést, jsou prováděny zde. Logická vrstva tvoří mezivrstvu mezi prezentační a datovou vrstvou, se kterými komunikuje pomocí rozhraní. Tato vrstva má na starosti balíčky:

Managers – balíček `managers` obsahuje třídy, které zprostředkovávají komunikaci mezi prezentační a datovou vrstvou a zároveň se starají o business logiku aplikace.

Custom vendor – balíček `custom vendor` je balíček, ve kterém se nacházejí komponenty z externích projektů, které nelze stáhnout za pomoci *Composeru*.

Security – balíček `security` je balíček obsahující třídy, které mají na starost autentizaci, autorizaci a kontrolu přístupů uživatele.

2.1.3 Datová vrstva

Datová vrstva obsahuje třídy, které umožňují provádět CRUD operace nad datovým úložištěm. Vrstva zároveň definuje rozhraní, které umožní logické vrstvě tyto operace provádět. Tato vrstva se skládá z následujících balíčků:

Entities – `entities` je balíček, který obsahuje veškeré entity, se kterými aplikace pracuje. Tento balíček používají veškeré balíčky, které pracují s daty z datového úložiště, včetně vrstvy prezentační. Třídy obsahují definici, metody a specifikaci, jak se má konkrétní třída mapovat na tabulky v databázi.

DAO – balíček `dao` obsahuje třídy komunikující s konkrétním datovým úložištěm.

2.2 Databázový model

Databázový model je stejně jako doménový model zobrazen pomocí UML[1] diagramu. Od doménového modelu se databázový liší v tom, že obsahuje konkrétní informace o atributech, vztazích, cizích klíčích a implementaci databáze. Model pomáhá programátorům k lepší orientaci v dané struktuře aplikace. Všechny tabulky mají prefix „jr_“, aby bylo na první pohled poznat, že se jedná o tabulky pro aplikaci Journal.

V této kapitole budou popsány pouze změny, které byly provedeny v rámci bakalářské práce. Zbylé tabulky nebudou vysvětleny, protože většina jich je samovysvětlujících. Na modelu na obrázku 2.2 je zobrazen stav struktury, který je finální po provedení všech změn.

Databázový model byl stejně jako doménový model rozšířen pouze o pár informací. Hlavní změnou je přidání tabulky `jr_osoba`, která je důležitá pro označování a spravování osob u událostí. Druhá velká změna je v přidání tabulky `jr_terms`, která poslouží pro uchování jednotlivých smluvních podmínek jak již bylo vysvětleno v sekci 1.1.4.

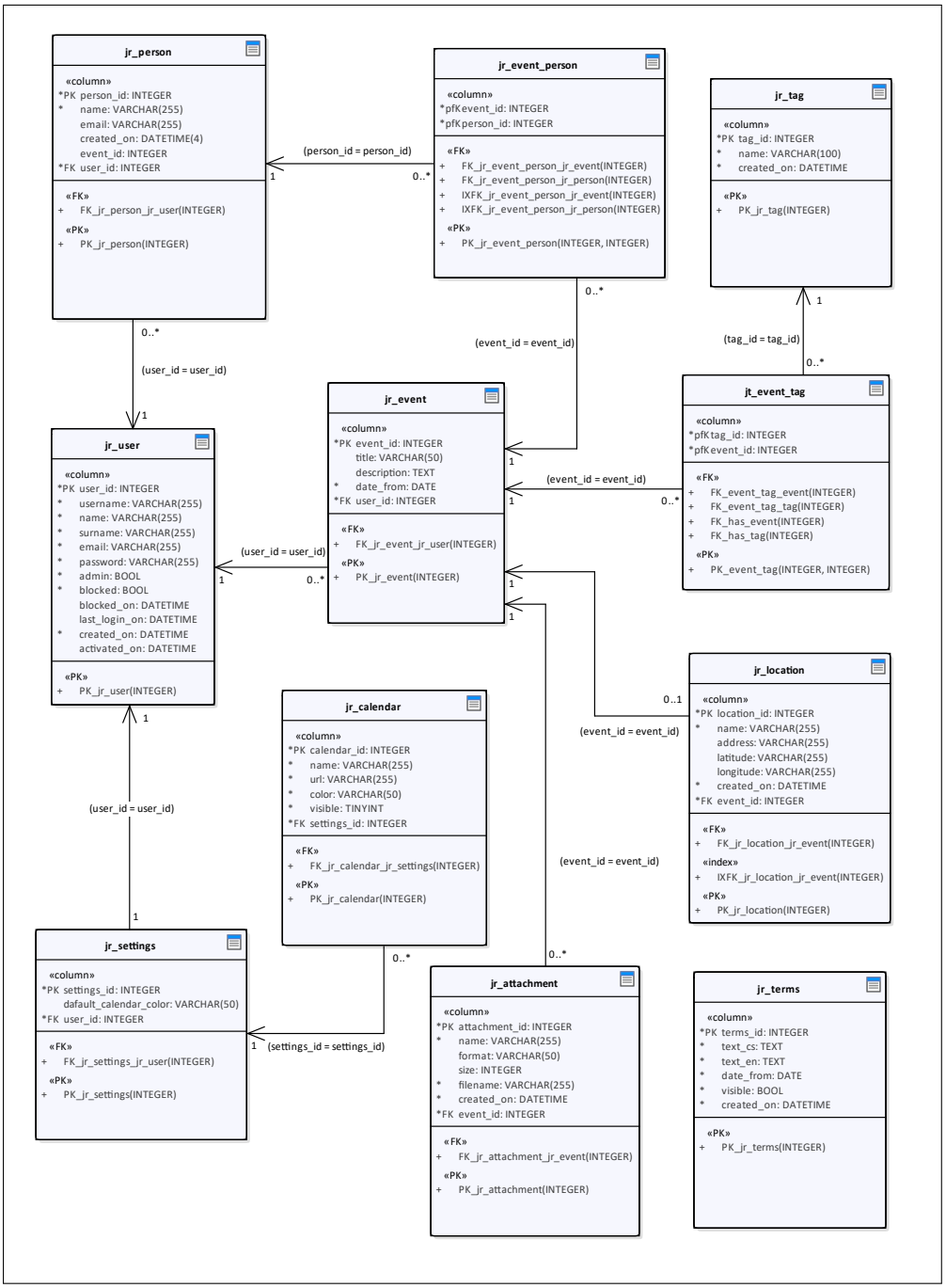
Poté jsou přidány a odebrány některé atributy v původních tabulkách. Přidán byl například atribut `admin` a `last_login_on` k tabulce `jr_user`. Pro tabulku `jr_person` byla přidána vazba *Many-To-Many* mezi tabulkou `jr_user` a `jr_person`. Jako další byla přidána rekurzivní vazba *One-To-Many* nad tabulkou `jr_event`, která bude mít na starost hierarchii událostí. Implementace této vazby je popsána v další kapitole.

2.3 Návrhový model tříd

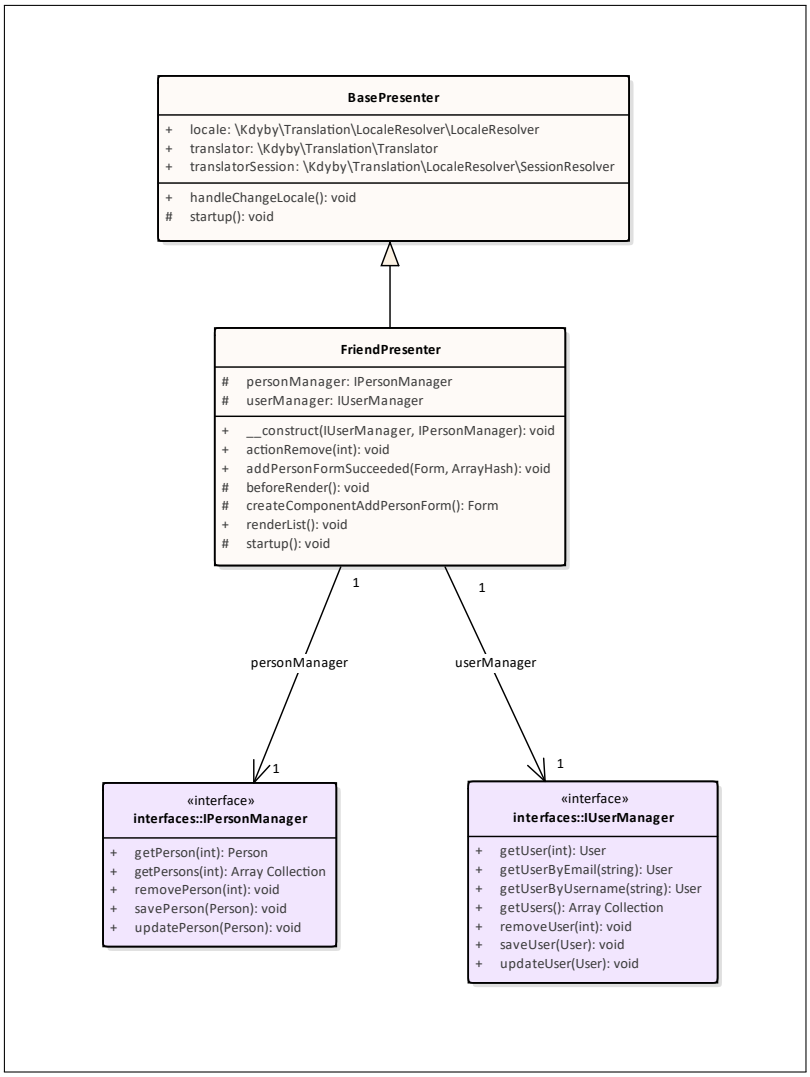
Tato sekce obsahuje detailní popis aplikace. Jsou zde detailně popsány hotové třídy, ze kterých se aplikace skládá, včetně proměnných a metod. Názvy tříd, složek a atributů odpovídají názvům použitých přímo v aplikaci. Aplikace se skládá ze tří vrstev. Jednotlivé vrstvy mezi sebou komunikují za pomoci rozhraní. Každá vrstva je navržena tak, aby byla schopna pracovat nezávisle na ostatních.

Diagramy tříd jsou jedny z nejvíce používaných UML diagramů, zobrazující objekty implementované v systému a jejich vzájemné vazby. [1] Jelikož návrhový model tříd byl již popsán v rámci předmětů BI-SP1 a BI-SP2, tak obrázky s diagramy 2.3, 2.4 a 2.5 budou zaměřeny pouze na nový `FriendPresenter`, který se stará o zobrazení, přidání, smazání osob a zpracování formuláře na přidání osob.

Velká část tříd v aplikaci rozšiřuje základní objekty *Nette* frameworku [3], proto nejsou tyto objekty zachyceny v diagramech, neboť cílem je popsat pouze řešení aplikace, nikoli knihoven *Nette* frameworku.



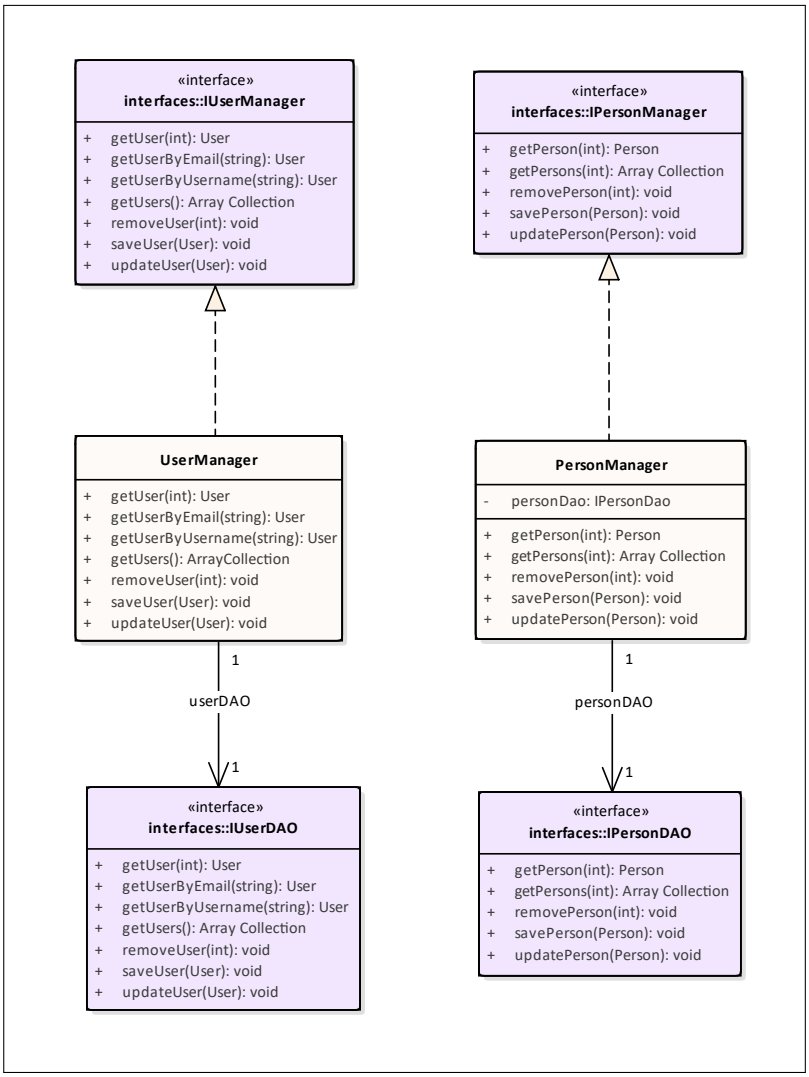
Obrázek 2.2: Databázový model aplikace



Obrázek 2.3: Třídní model se zaměřením na prezentační vrstvu

2.3.1 BasePresenter

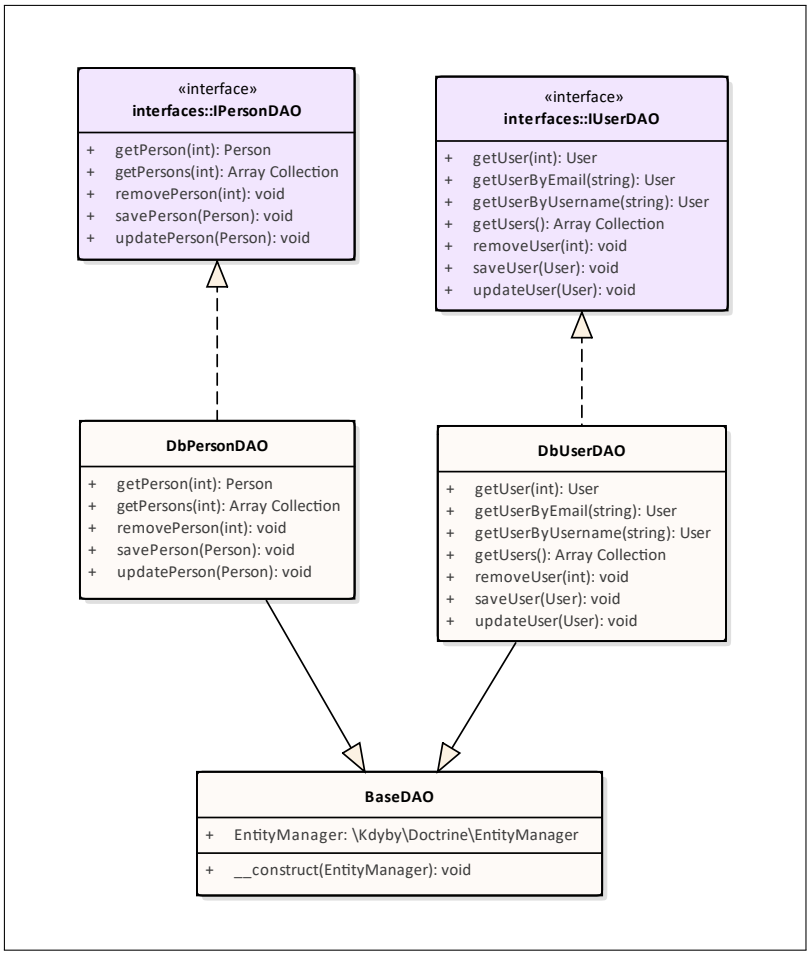
`BasePresenter` je rodič všech Presenterů v aplikaci, který obsahuje často používané metody. A také pro šablonu připravuje proměnné, které se používají v každé šabloně. Metoda `startup()` ověřuje před provedením akce potomka, zda má uživatel dostatečné oprávnění pro provedení dané akce. Ještě se zde nalézá metoda `handleChangeLocale($locale)`, která se společně se třemi proměnnými `locale`, `translator` a `translatorSession` stará o změnu jazyka v rámci celé aplikace.



Obrázek 2.4: Třídní model se zaměřením na logickou vrstvu

2.3.2 FriendPresenter

`FriendPresenter` je třída, která rozšiřuje `BasePresenter` a stará se o vytváření, upravování, zobrazení a mazání osob. Obsahuje metodu `renderList()` starající se o vytvoření stránky se seznamem osob. Poté jsou zde dvě metody, `createComponentAddPersonForm()` a `addPersonFormSucceded()`, které se starají o vytvoření formuláře a zpracování dat z něho. Jako poslední je zde metoda `actionRemove()` pro smazání osoby.



Obrázek 2.5: Třídní model se zaměřením na datovou vrstvu

2.3.3 PersonManager

Tato třída pro osoby nedělá žádné výpočty, pouze zprostředkovává komunikaci mezi třídami `FriendPresenter` a `DbPersonDAO` pro entitu `Person`.

2.3.4 DbPersonDAO

V aplikaci slouží všechny `DAO` třídy k manipulaci s databází za pomoci knihovny *Doctrine 2* [5]. Jelikož všechny `DAO` třídy mají specifikovaný svůj `interface`, není problém rozšířit či změnit nynější způsob práce s daty, bude-li toto rozhraní dodrženo. Každá `DAO` třída rozšiřuje `BaseDAO` třídu, která v konstruktoru obdrží `EntityManager`, za jehož pomoci manipulace s daty probíhá.

2.3.5 UserManager

Třída `UserManager` se stará o komunikaci s třídou `DbUserDAO` a slouží pro získání uživatele/uživatelů podle `id`, `username` nebo `email`. Také obsahuje metody `saveUser(), updateUser(), removeUser()`, které slouží pro uložení nového uživatele, upravení či smazání stávajícího uživatele.

2.3.6 DbUserDAO

Třída `DbUserDAO`, stejně jako třída `DbPersonDAO`, slouží k manipulaci s uživatelskými daty mezi aplikací a databází za pomoci *Doctrine 2* knihovny. Pro získání uživatelských dat existuje více funkcí, které získávají a upravují data podle různých parametrů viz třída `UserManager` 2.3.5.

2.3.7 BaseDAO

Třídu `BaseDAO` rozšiřuje každá `DAO` třída aplikace. V této třídě se vytvoří instance `EntityManager`, starající se o manipulaci dat za pomoci knihovny *Doctrine 2*.

2.4 Model komunikace

Pro úplné představení architektury je zde model komunikace na obrázku 2.6, který je znázorněn za pomoci UML sekvenčního diagramu [1]. Na diagramu je zachycen stejný případ jako v Návrhovém modelu tříd 2.3, a to „Přidání osoby“. Průběh tohoto procesu je následující:

1. Uživatel aplikace přistoupí na stránku „Seznam přátel“.

2. Tím se ve třídě `FriendPresenter` zavolá metoda `renderList()`, která zavolá metodu `getPersons(userId)` ze třídy `PersonManager`, která získá záznam o osobách ze třídy `DbPersonDAO`. Tyto osoby jsou předány zpátky metodě `renderList()`.

3. Zároveň se vytvoří formulář `createComponentAddPersonForm()` pro přidávání a úpravu osob. Společně se seznamem osob vykreslí data do šablony.

4. Uživatel poté ve formuláři vyplní data o přidávané osobě a formulář uloží (odešle).

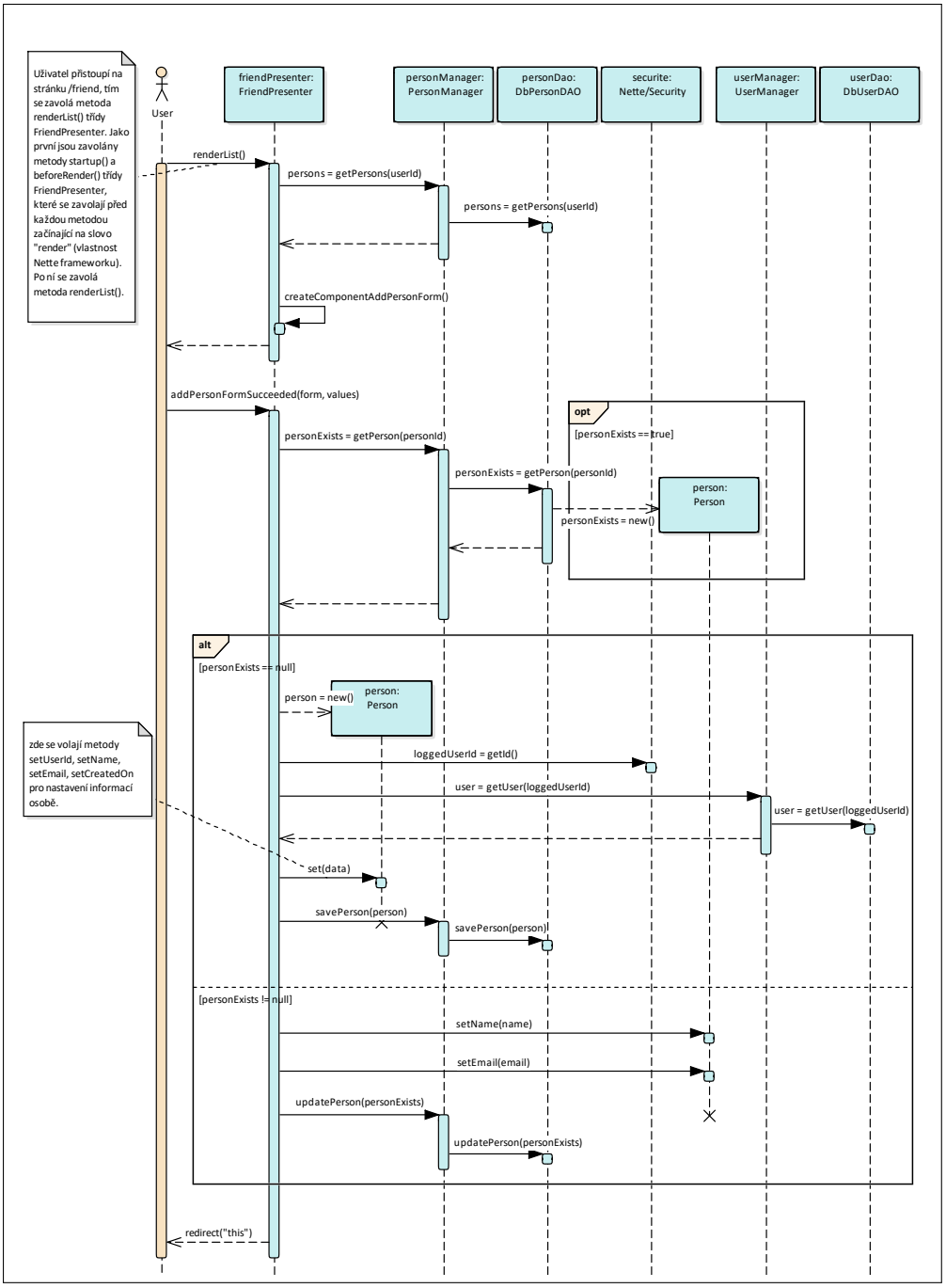
5. Uložením formuláře se zavolá `addPersonFormSucceeded(form, values)` metoda, která se stará o zpracování formuláře.

6. V této metodě se zkontroluje, zda tato osoba je nová nebo se jedná o osobu, která již existuje a tudíž je pouze upravována. To se zjistí voláním metody `getPerson(personId)`, která vrátí existující osobu, pokud již existuje.

7. Pokud osoba neexistuje, jedná se o přidávání nové osoby. Vytvoří se nová instance třídy `Person`, které se pomocí `set` metod přidají data o osobě jako název, email, datum vytvoření a uživatel, který ji vlastní. Poté je osoba uložena do databáze pomocí metody `savePerson(person)`.

8. Pokud osoba již existuje, pouze se jí změní název a email na data zadaná ve formuláři a uloží se za pomoci metody `updatePerson(personExists)`.

9. Nyní je uživatel přesměrován na stejnou stránku pouze s nově přidanou-/upravenou osobou. A proces přidání osoby je ukončen.



Obrázek 2.6: Sekvenční diagram znázorňující přidání osoby

Kapitola 3

Realizace a testování

Poslední je realizace kroků, které byly analyzovány a navrhnuty v předešlých dvou kapitolách této bakalářské práce. Budou zde rozebrány kroky, jak byla splněna realizace požadavků a problémy, které se během vývoje aplikace *Journal* vyskytly. Na konci kapitoly jsou popsány manuály a testování aplikace. Po realizaci celé bakalářské práce byla aplikace nasazena na veřejný server, je tedy dostupná na adrese `http://journal.jecool.net`.

3.1 Realizace nových požadavků

Jelikož bylo během bakalářské práce uděláno mnoho změn na aplikaci. Bude zde představena pouze realizace čtyř hlavních požadavků na aplikaci, které byly stanoveny v analytické kapitole 1.3.1.1.

3.1.1 Lokalizace

Na každé stránce aplikace je v sidebaru možnost přepnout jazyk aplikace z češtiny do angličtiny a naopak. Při změně jazyka je stránka pouze obnovena s novým jazykem a uživatel tak zůstane na původní stránce. Při odesílání emailu uživateli (aktivační email / nové heslo) je email přeložen podle jazyka, ve kterém uživatel prohlíží aplikaci.

Překlad aplikace je řešen pomocí knihovny *Kdyby/Translation* [24], která poskytuje integraci *Symfony/Translation* pro *Nette* framework [3]. Instalace této knihovny proběhla za pomoci *Composer*. Poté bylo potřeba v konfiguračním souboru `config.neon` přidat knihovnu jako extension viz ukázka 3.1.

```
extensions:
    translation: Kdyby\Translation\DI\TranslationExtension
```

Výpis kódu 3.1: Ukázka extension v souboru `config.neon`

Dále bylo v `BasePresenteru` potřeba přidat proměnnou `$locale` a službu `$translator`, ta se stará o překlad a službu `$translatorSession`, která má na starost přepínání jazyka. Proměnná `$locale` je persistentní, aby se během procházení webu neztratila. V `BasePresenteru` byla dále přidána funkce `handleChangeLocale($locale)`, která se volá pro přepnutí jazyka. Vše je zobrazeno v ukázce 3.1.1.

```
abstract class BasePresenter extends Presenter
{
    /** @persistent */
    public $locale;

    /** @var \Kdyby\Translation\Translator @inject */
    public $translator;

    /** @var \Kdyby\Translation\LocaleResolver\
       SessionResolver @inject */
    public $translatorSession;

    public function handleChangeLocale($locale)
    {
        $this->translatorSession->setLocale($locale);
        $this->redirect('this');
    }
}
```

Výpis kódu 3.2: Ukázka třídy `BasePresenter`

Následně bylo třeba vytvořit slovníky. Každá stránka má svojí složku, ve které jsou vždy dva soubory, jeden s českým a jeden s anglickým překladem textů. Soubor s českým překladem na ukázce kódu 3.3 má stejný název jako *Latte* šablona stránky, ale má koncovku *.cs_CZ.neon*, což značí, že jde o český jazyk. Soubor s anglickým překladem na ukázce 3.4 má stejný název souboru jako český překlad ale má koncovku *.en_US.neon*.

V Latte šablonách se pak používají makra pro volání překladu ze slovníku ve tvaru `{_ název_souboru.název_proměnné}`. V šabloně, která používá slovníky z ukázek kódů 3.3, 3.4 by se překlad pro title volal `{_ list.title}` a podle toho, v jakém jazyce uživatel aplikaci používá by se zvolil příslušný slovník. [25]

Pro překlad textu, který se překládá již v třídách `presenter` slouží proměnná `$translator` která, jak již bylo psáno, je definována v `BasePresenter`. Překlad probíhá pomocí metody `translate($message)`, které se předá `string` s *název_slovníku.název_proměnné* stejně jako v *Latte* šabloně. Volání příkazu pro již zmiňovaný `title` by vypadalo takto:

```
$this->translator->translate('list.title');
```
[26]

```
title: Úvodní stránka
deleteConfirm: Opravdu chcete událost odstranit?
deleted: Událost byla odstraněna
filterForm:
    name: Název události
    tags: Tagy
    date: Datum událostí
    person: Osoba
    search: Vyhledat události

table:
    name: Název
    tags: Tagy
    date: Datum události
    location: Lokace
    persons: Osoby
    noFound: Nenalezeny žádné události
    noEvents: Nemáte vytvořeny žádné události
```

Výpis kódu 3.3: Ukázka českého překladu `list.cz_CS.neon`

```
title: Home
deleteConfirm: Do you want to delete this event?
deleted: Event was successfully deleted
filterForm:
    name: Name of event
    tags: Tags
    date: Date of event
    person: Person
    search: Search events

table:
    name: Name
    tags: Tags
    date: Date of event
    location: Location
    persons: People
    noFound: No events found
    noEvents: No events created
```

Výpis kódu 3.4: Ukázka anglického překladu `list.en_US.neon`

3.1.2 Označování osob

Nové osoby lze přidávat na stránce s osobami, kde je lze také upravovat a mazat. Nová osoba se však vytvoří i v případě, že ještě neexistuje a uživatel ji označí u události. Ze seznamu osob se lze také přes tlačítko *zobrazit události* dostat na seznam událostí, ve kterých je osoba označena. Uživateli se označené osoby u události zobrazují v pravém sidebaru v detailu události, odkud se lze kliknutím na název osoby dostat také na seznam událostí, ve kterých je osoba označena.

Označování osob při vytváření nové události je zajištěno pomocí knihovny *Tagify* [27]. Tato knihovna je nastavena tak, aby osoby po třech napsaných písmenech našeptávala. Přidávání jedné osoby se ukončí znakem čárka nebo enterem. Po ukončení přidávání osoby se osoba obalí rámečkem. Po najetí na osobu se zobrazí křížek, kterým se dá osoba z přidávání odebrat. Přidáli uživatel dvě stejné osoby, tak ta poslední se vizuálně sama odstraní. Při upravování události se již uložené osoby načtou a zobrazí se obalené rámečkem s křížkem pro smazání.

Knihovna *Tagify* vrací osoby ve formátu *JSON* (JavaScript Object Notation), který se následně v třídě `EventBuilder` rozparsuje a k události se přidají jednotlivé osoby. Při přidávání osob k události se pak rozlišuje, zda již osoba existuje, nebo se vytvoří nová, která se přidá.

3.1.3 Monitorování aplikace

Pro využití služby monitorování aplikace je potřeba mít administrátorská práva. Tato práva lze přidat přímo v databázi nebo od jiného administrátora. Monitorování se skládá ze čtyř stránek.

První se jmenuje „Monitorování aktivity“, zde má administrátor přehled nových uživatelů (rozdělených podle registrace na dnes, tento týden, tento měsíc a celkem) a přehled nových událostí (rozdělených stejně jako informace o uživatelích). Dále je zde zobrazen počet blokovaných a neaktivovaných uživatelů. Pro lepší přehlednost je zde graf s počtem přidaných událostí za poslední týden. Pro zajímavost je vypsán počet příloh a jejich velikost, externích kalendářů a tagů všech uživatelů.

Druhá stránka pod názvem „Práva uživatelů“ zobrazuje seznam uživatelů registrovaných v aplikaci. Každému uživateli zde lze měnit práva jako administrátor, aktivován a blokován. Dále je zde zobrazen email a datum posledního přihlášení. Když je uživatel zablokován, je navíc vypsáno datum, od kdy je blokovaný. Je zde také možno uživatelský účet zcela odstranit. Pro odstranění je potřeba projít přes ochranou podmínku a to je zadání hesla. V seznamu uživatelů lze vyhledávat podle uživatelského jména, příjmení, emailu, zda je administrátor či zda je blokován. Vyhledávání zda je administrátor je vyře-

šeno selectboxem. V selextboxu je na výběr „nezvoleno“, „je administrátor“ a „není administrátor“. To samé platí i pro selectbox blokován. Vztah mezi jednotlivými vyhledávanými atributy je při vyhledávání v databázi `AND`.

Na této stránce je také tlačítko „Odeslat email všem“ které zobrazí stránku pro odeslání emailu všem uživatelům aplikace. Na této stránce se vyplní zpráva a předmět emailu a odešle se všem uživatelům registrovaným v aplikaci.

Dále je zde stránka „Smluvní podmínky“ v administrátorsté části aplikace, kde má administrátor přehled všech přidaných smluvních podmínek. Může zde přidávat nové podmínky, upravovat stávající či nastavit jako aktuální podmínky jiné.

3.1.4 Hierarchie událostí

Hierarchické uspořádání událostí bylo rozděleno na dvě úrovně (rodič, potomek). Hlavní událost (dále „rodičovská“) událost vznikne z normální jednoúrovňové události přidáním podudálosti (dále „dětské“). Přidat dětskou událost k hlavní události lze z detailu rodičovské události, či ze seznamu událostí. Smazat rodičovskou událost lze, pouze pokud nemá žádné dětské události, to znamená, je to normální jednoúrovňová událost. Při pokusu o smazání rodičovské události je uživateli zobrazena hláška, že událost má stále dětské události, a proto ji nelze smazat.

Pro hierarchické uspořádání bylo potřeba přidat novou rekurzivní vazbu pro třídu `Event`. Jedná se o vazbu *One-To-Many* mezi proměnnými `$parentID` a `$children`. Mapování rekurzivní vazby *One-To-Many* je zobrazeno na ukázce kódu 3.5.

3.2 Řešené problémy

V průběhu realizace bakalářské práce bylo napsáno mnoho nových funkcionalit. V této sekci bude představeno několik zajímavých věcí, které bylo potřeba vyřešit.

3.2.1 Routování

První problém, který bylo potřeba vyřešit byla změna routování aplikace pro lokalizaci, aby bylo směrování *HTTP* požadavků na správné *presenter* třídy. Routování již bylo řešeno v předchozím vývoji aplikace, kde se použil doporučený způsob podle *Nette* dokumentace, a to vytvoření `RouteList` v metodě `createRouter()` třídy `RouterFactory`. Zde se vytvoří `RouteList` ze dvou jiných, jedna má na starosti *REST API* pro mobilní aplikaci, druhá se stará o přesměrování na správný *presenter* webové aplikace.

```
/**
 * @ORM\OneToMany(targetEntity="Event",
                   mappedBy="parentId",
                   cascade={"persist", "remove"})
 * @var Event[]
 */
private $children;

/**
 * @ORM\ManyToOne(targetEntity="Event",
                  inversedBy="children")
 * @JoinColumn(name="parent_id",
                referencedColumnName="event_id",
                onDelete="SET NULL")
 */
private $parentId;

public function __construct()
{
    $this->children = new ArrayCollection();
}
```

Výpis kódu 3.5: Ukázka rekurzivního mapování událostí

Problémem bylo nové přidání lokalizace pouze pro webovou aplikaci, která se zapisuje do *URL* jako `cs` pro český jazyk nebo `en` pro anglický jazyk. Bylo tedy nutné rozšířit původní `RouteList` o typ jazyku. K rozšíření došlo pomocí volitelné masky *<locale>* v cestě routeru. Český jazyk byl nastaven jako výchozí a anglický jako možný volitelný, jak je vidět v ukázce kódu 3.6.

3.2.2 Kalendář

V průběhu implementace bakalářské práce bylo zjištěno, že kalendář v aplikaci nefunguje tak, jak by měl. Problém byl s načítáním externích kalendářů. Data se v kalendáři buďto vůbec nezobrazovala, nebo kalendář vůbec nešel přidat, jelikož bylo špatně navržené parsování událostí z externích kalendářů a nepočítalo s neobvyklými případy událostí jako událost s nulovým časem nebo událost se speciálními znaky v názvu či popisu. Bylo tedy nutné poupravit kód, který z kalendáře vytváří *JSON* objekt pro knihovnu *FullCalendar* [28], pomocí které je zobrazen kalendář a události v něm.

Po implementaci tohoto řešení se pokračovalo dál v práci na kalendáři, jelikož byly nalezeny další chyby. Jako další chyba se projevily externí události s „nulovým časem“, kdy taková událost nebyla správně načtena a proto se v kalendáři nezobrazovala. K dalším chybám patří například speciální znaky v názvu či popisu externích událostí. Po implementaci těchto problémů se pokračovalo dál v práci na kalendáři, jelikož se objevily další nedostatky.

V kalendáři bylo přepracováno vypínání externích kalendářů, které do této doby bylo možné pouze v nastavení. Nyní lze kalendáře vypínat a zapínat přímo na stránce kalendáře, kde se změny projevují ihned. Největší problém byl ale s opakujícími se událostmi externích kalendářů. Knihovna *FullCalendar*, která byla pro zobrazení kalendářů použita od začátku vývoje aplikace, nepodporuje zobrazování externích kalendářů.

```
/** Control routing for the whole application
 *   @package App\Router
 */
class RouterFactory
{
   /** Create Router for the whole application
    *   @return RouteList
    */
   public static function createRouter()
   {
       $router = new RouteList();

       $router[] = new Route('Error/<action>',
                             'Error:default');

       $router[] = $restRouter = new RouteList('Rest');
       $restRouter []= new CrudRoute('rest/<presenter>
         [/<id>[/<relation>[/<relationId>]]]');

       $router[] = $frontRouter = new RouteList( 'Front');
       $frontRouter[] = new Route('activate/<hash>',
         'Activation:activate');
       $frontRouter []= new Route('[<locale=cs cs|en>/]
          settings/',
         'Setting:default');
       $frontRouter []= new Route('[<locale=cs cs|en>/]
         <presenter>/<action>/[<id>]', 'Event:list');

       return $router;
   }
}
```

Výpis kódu 3.6: Ukázka `RouteFactory` třídy

Bylo by tedy nutné přepracovat celý kalendář a použít jinou knihovnu. Po delším hledání, kdy nebyla nalezena žádná dostačující knihovna, která by mohla nahradit nynější, bylo zjištěno, že *FullCalendar* připravuje novou verzi, která bude mít plugin s podporou pro *RRULE*[4], který umožní opakující se události lehce zobrazit. Bylo tedy rozhodnuto, že v aplikaci zůstane nynější kalendář,

[4]textový formát s popisem opakování události. Ukázka: RRULE:FREQ=WEEKLY; COUNT=30; INTERVAL=1; WKST=MO

který po vydání nové verze knihovny *FullCalendar* bude aktualizován a upraven o zobrazování opakujících se aplikací. Již nyní bylo předávání informací ve formátu *JSON* rozšířeno o informaci o opakování události, aby pozdější implementace nebyla tak rozsáhlá. Aktuálně se tedy opakující událost v kalendáři zobrazí pouze v první den výskytu.

3.2.3 Mapa

Při vývoji této bakalářské práce bylo zjištěno, že na mapovém přehledu nelze zobrazit a rozkliknout všechny markery. Problém byl u událostí nacházejících se na stejném místě, jelikož se shlukovali a i po maximálním přiblížení mapy se nezobrazili a zůstalo pouze kolečko s počtem shluknutých událostí. Nebylo tedy možné si tyto události rozkliknout a zobrazit si jejich detail.

Tento problém byl vyřešen pomocí sloupce s událostmi, který se zobrazuje vedle mapy. V tomto sloupci jsou zobrazeny všechny události, které jsou zrovna viditelné v rámečku mapy, k tomuto řešení mne inspirovala ukázka kódu [29]. Takže po plném přiblížení na události, které jsou na stejném místě, jsou v seznamu událostí zobrazeny jen a pouze ty události, které jsou shluknuty v kolečku. Stále je nelze rozkliknout na mapě, ale v seznamu je vidět název a datum všech událostí, které se na tomto místě děly. Z tohoto seznamu se lze prokliknou přes jednotlivé události na jejich detail.

3.3 Verzování

V průběhu vývoje aplikace je potřeba mít přehled o změnách, které se postupem času prováděly, a také umožnit pracovat více lidem současně na jednom projektu. Celý projekt byl tedy verzován. V průběhu bakalářské práce na aplikaci sice pracoval pouze jeden člověk, ale i tak je výhoda verzovat kód. Možností, jak verzovat je několik, například SVN, CVS (Concurrent Version System) či Git, který je v dnešní době nejužívanější a je velmi spolehlivý.

Tento projekt je verzován pomocí SVN a Git zároveň, jelikož každá distribuce má něco, co je pro projekt důležité. Pomocí SVN je verzovaná veškerá dokumentace, která byla vytvořena pomocí aplikace *Enterprise Architect*. Samostatný kód aplikace je poté verzován pomocí Git. Oba projekty jsou ale verzovány v soukromých projektech, a proto nejsou přístupné pro veřejnost.

3.4 Manuály aplikace

Nová verze aplikace obsahuje tři manuály. Tyto manuály vycházejí z příruček, které byly vytvořeny v předchozím běhu projektu. Do příruček byly přidány nové informace a návody, zároveň některé kroky byly odebrány, jelikož již

neplatí. Veškeré manuály jsou k bakalářské práci dodány jako elektronická příloha.

3.4.1 Uživatelská příručka

Uživatelská příručka slouží k seznámení nového uživatele se všemi dostupnými funkcionalitami a důležitými pojmy aplikace. Existují dva typy uživatelské příručky:

uživatelská – uživatelská příručka je přístupná pro každého uživatele.

administrátorská – administrátorská příručka je přístupná pouze pro administrátory v aplikaci.

Aby byla příručka pro každého uživatele k dispozici, tak je umístěna přímo v samotné aplikaci. K příručce mají přístup i neregistrovaní uživatelé, kterým může pomoci se rozhodnout, zda jim aplikace vyhovuje a chtějí si založit účet.

3.4.2 Vývojářská příručka

Vývojářská příručka je popis kroků, potřebných pro to, aby si mohl nový programátor aplikaci stáhnout a dále rozšiřovat. Jsou zde popsány technologie, použité při vývoji aplikace. Dále je zde popsáno verzování, struktura aplikace a další důležité kroky pro zprovoznění.

3.4.3 Příručka pro nasazení

Tato příručka slouží pro pouhé nasazení již hotového release balíčku na server. V příručce je podrobný návod, jak celou aplikaci nainstalovat, připravit databázi a aplikaci nakonfigurovat z release balíčku 1.0, který byl vydán s touto bakalářskou prací. Je zde také návod, jak aplikaci aktualizovat z již existujícího balíčku 0.4, který byl vydán jako poslední v předmětu BI-SP2.

3.5 Testování

Poslední velmi důležitou částí je testování, sloužící ke zjištění, zda aplikace funguje jak má. To znamená, že vše co bylo implementováno funguje a uživatel nedostane chybné hlášky. *Journal* využívá dvou způsobů testování.

3.5.1 Unit testy

Jedná se o kategorii testů, které již byly v malé míře v aplikaci vytvořeny. Při rozšiřování aplikace bylo ale potřeba udělat mnoho nových testů a doplnit ty původní. Tyto testy mají na starost testovat jednotlivé metody jedné třídy. Všechny závislosti a volání dalších tříd jsou nahrazeny „mock“ třídami. Tyto

testy se nacházejí v jádru aplikace ve složce `tests` a jsou rozděleny do složek `FrontModule` a `RestModule`, stejně jako celá aplikace.

Tyto testy jsou cíleny na prezentační a logickou vrstvu aplikace. V současné době jsou pokryty následující třídy:

- Event
- User
- CalendarBuilder
- EventBuilder
- UserBuilder
- AdministrationPresenter
- AuthenticationPresenter
- EventPresenter
- FriendPresenter
- SettingPresenter

3.5.2 Uživatelské testování

Jde o scénáře, pomocí kterých si může vývojář či testující osoba odzkoušet, zda v aplikaci funguje vše správně. Testy se skládají ze dvou scénářů, jde o scénář pro uživatele a o scénář pro administrátora. Scénáře vypadají následovně:

1. **Uživatelský scénář** popisuje testování od registrace přes všechny funkcionality aplikace

   a) **Registrace nového uživatele** – zkusit se zaregistrovat s chybnými údaji (chybějící email / email, který nemá emailovou formu / neshodující se hesla / krátká hesla / již existující uživatelské jméno), na toto by aplikace měla upozornit chybou a vyzvat uživatele k opravení daného problému. Poté se zaregistrovat se správnými údaji.

   b) **Zkouška přihlášení** – pokusit se přihlásit na nově vytvořený účet. Aplikace by měla vypsat hlášku „Účet nebyl aktivován, aktivujte jej pomocí odkazu v emailu“.

   c) **Aktivace účtu** – pomocí emailu, který byl obdržen po registraci aktivovat účet, aplikace by měla vypsat informaci „Účet byl úspěšně

aktivován!“. Poté zkusit účet aktivovat znovu, měla by být vypsána hláška „Účet již byl aktivován!“.

d) **Zapomenuté heslo a jeho změna** – vyzkoušet formulář pro získání zapomenutého hesla (s neplatnými údaji). Poté požádat o heslo se správnými údaji a po obdržení emailu kliknout na odkaz, který zobrazí stránku pro zadání nového hesla. Zde znovu vyzkoušet nevalidní vstupy (neshodující se hesla / krátká hesla / heslo bez čísel, malých a velkých písmen). Poté zadat validní heslo a změnit ho.

e) **Přihlášení** – zkusit se přihlásit s nevalidními údaji. Aplikace by měla zobrazit hlášku, že je zadáno špatné jméno nebo heslo. Poté se přihlásit do aplikace s validními údaji.

f) **Přidání události** – vytvořit několik různých událostí s názvem, popisem, různými tagy (i opakujícími se), datem, přílohami, více osobami (jedno jmennými i více jmennými) a lokací. Po přidání by se měl zobrazit detail nové událost s hláškou „Událost byla úspěšně uložena“. U události s přílohami zkusit smazat přílohu. Při mazání by se mělo zobrazit modálové okno, zda chcete přílohu opravdu smazat. Po smazání má být vypsána hláška „Příloha byla úspěšně smazána“.

g) **Upravení události** – zkusit upravit událost, kde změnit všechny údaje. Po uložení by se měl zobrazit detail nově upravené události s hláškou „Událost byla úspěšně uložena“.

h) **Smazání události** – smazat událost z detailu události i ze seznamu událostí. Vždy by mělo být zobrazeno potvrzovací modálové okno s dotazem, zda chcete událost skutečně odstranit. Po odstranění události by měla být vypsána hláška „Událost byla odstraněna“.

i) **Vyhledávání události** – vyhledat událost/události podle jména, data, tagu/tagů či osoby v různých kombinacích.

j) **Změna hesla v nastavení** – v nastavení aplikace zkusit změnit heslo s nevalidními vstupy (špatné aktuální heslo / neshodující se nová hesla / hesla bez malých písmen, velkých písmen, čísel). Poté změnit heslo s validními údaji. Po změně hesla se odhlásit z aplikace a zkusit se přihlásit znovu se starým heslem a poté se správným novým heslem.

k) **Přidání externího kalendáře** – v nastavení importovat nějaký google kalendář.

l) **Zobrazení událostí v kalendáři** – zobrazit si události v kalendáři a jednotlivé kalendáře zkoušet vypínat, zda se vypínají ty správné.

Poté si zkusit rozkliknout různé události v kalendáři, zda se zobrazují správná modálová okna. Zkusit přejít na detail některé existující události v *Journal* kalendáři.

m) **Přidání externí události** – v kalendáři v modálovém okně externí události dát „Přidat událost“ a zkontrolovat, zda se zobrazila stránka pro přidání události s již správně předvyplněnými daty z externí události (název, popisek, datum).

n) **Přidání nové osoby** – na stránce s osobami zkusit přidat novou osobu.

o) **Upravení a odstranění osoby** – zkusit upravit již existující osobu (změnit název / email). Poté některou osobu zkusit odstranit. Mělo by se zobrazit modálové okno s dotazem, zda chcete osobu smazat.

p) **Zobrazení událostí s osobou** – ze seznamu osob si zkusit zobrazit události s osobou za pomoci tlačítka s ikonou oka.

q) **Zobrazení událostí na mapě** – zobrazit si události na mapě. Otestovat, zda se v pravém sloupci vypisují správné události. Vyzkoušet, zda se vypisují i události se stejným místem. V mapě vyzkoušet „shlukování událostí“ do koleček s počtem událostí.

r) **Přidat podudálost** – ze seznamu událostí zkusit k některé události přidat podudálost. Stejně tak zkusit přidat podudálost události z detailu události. Zkontrolovat, že nelze přidávat podudálost již vzniklým podudálostem.

s) **Kontrola hierarchie událostí** – na seznamu událostí zkontrolovat, zda se zobrazují správné ikonky u rodičovských a dětských událostí (rodičovská událost má ikonu „sítě/větvení“ a podudálost má ikonu „sdílení/řetězu“). V detailu rodičovské události zkontrolovat, zda se správně zobrazuje seznam podudálostí. Prokliknout se na detail některé z podudálostí a zkontrolovat, zda se pod názvem události zobrazuje název hlavní události. Zkusit se prokliknout na detail hlavní události přes název hlavní události. V kalendáři zkontrolovat, zda se podudálosti zobrazují světlejší zelenou barvou.

t) **Změna jazyka** vyzkoušet, zda funguje přepínání jazyka. Tento celý test nejlépe opakovat i v nově zvoleném jazyce.

2. **Administrátorský scénář** počítá s již existujícím administrátorským účtem, který byl vytvořen již z uživatelsky testovaného účtu.

a) **Zobrazení monitorování** – zobrazit stránku „Monitorování aktivity“ a zkontrolovat, zda aplikace zobrazuje data všude.

b) **Zobrazení práv uživatele** – zobrazit stránku s uživatelskými právy.

c) **Vyhledání uživatelského účtu** – vyhledat uživatele podle uživatelského jména, příjmení, emailu, zda je administrátor, zda je blokovaný a v různých kombinacích.

d) **Zablokovat účet** – najít účet k zablokování a pomocí kliknutí na ikonu ve sloupci „blokován“ si účet zablokovat. Zjistit, zda přišel na zablokovaný účet informační email a zkusit se přihlásit do aplikace se zablokovaným účtem. Aplikace by měla vypsat hlášku „Uživatelský účet je zablokovaný. Pokud si myslíte, že jde o nedorozumění, prosím kontaktujte administrátora“. Poté se přihlásit zpátky na administrátorský účet.

e) **Aktivace účtu** – zkusit si najít uživatele, který není aktivován a aktivovat ho pomocí kliknutí na ikonku ve sloupci „aktivován“.

f) **Změna účtu na administrátorský účet** – zkusit z některého uživatelského účtu udělat účet administrátorský pomocí kliknutí na ikonu ve sloupci „administrátor“. Zkusit se přihlásit s novým administrátorským účtem a zjistit, zda má administrátorské funkce.

g) **Odstranění uživatele** – najít uživatele a zkusit ho odstranit pomocí ikonky koše na pravé straně. Mělo by se zobrazit modálové okno s informací o mazání účtu a požadavkem na vaše heslo pro potvrzení smazání uživatele. Zadat heslo a smazat uživatele. Zkontrolovat, zda na smazaného uživatele přišel email s informací o smazání. Zkusit se přihlásit do aplikace se smazaným účtem. Poté se zpět přihlásit s administrátorským účtem.

h) **Změna jazyka** – vyzkoušet, zda funguje přepínání jazyka. Tento celý test nejlépe opakovat i v nově zvoleném jazyce.

3.5.3 Kompatibilita s prohlížeči

Aplikace je vyvíjena jako webová aplikace, poběží tedy na některém internetovém prohlížeči. Aplikace byla vytvořena a otestována na prohlížečích:

- Safari – version 12.1
- Google Chrome – version 73.0
- Firefox – version 66.0

Závěr

Cílem této práce bylo analyzovat nové funkční požadavky pro rozšíření aplikace a tyto požadavky implementovat. Nejdříve je zde popsána analýza výchozí podoby aplikace, na kterou navazuje analýza nových rozšíření a funkcionalit, poté byl vypracován návrh aplikace a následně byly realizovány nová rozšíření.

Všechny předem stanovené funkcionality byly splněny a výsledkem je nová verze aplikace, a to verze 1.0, včetně potřebných manuálů pro její chod. Aplikace není v perfektním stavu, stále je zde co zlepšovat (například zobrazení opakujících se událostí z externích kalendářů), ale i přes tyto nedostatky je aplikace ve funkčním stavu.

Finální verze této bakalářské práce v budoucnu může posloužit jako dobrý výchozí bod pro nové týmy či další bakalářské práce, které by aplikaci dále rozvíjely. Jako další by bylo možné rozšířit v aplikaci kalendář o správné zobrazování opakujících se událostí případně navrhnout a implementovat přidávání přátel v rámci aplikace či sdílení událostí pomocí odkazu přátelům a uživatelům, kteří nevyužívají aplikaci.

Literatura

[1] ARLOW, Jim a Ila NEUSTADT. UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky. 2. aktualiz. a dopl. vyd. Brno: Computer Press, 2007. [cit. 2019-04-20]. ISBN 978-80-251-1503-9.

[2] PHP: Hypertext Preprocessor. 2001 [cit. 2019-04-08]. Dostupné z: https://www.php.net

[3] Nette Foundation: 2008, c.:Rychlý a pohodlný vývoj webových aplikací v PHP | Nette Framework [online]. [cit. 2019-04-08]. Dostupné z: https://nette.org

[4] MySQL [online]. [cit. 2019-04-08]. Dostupné z: https://www.mysql.com

[5] Doctrine, The Open-Source PHP ORM and Persistence Tools Project [online]. [cit. 2019-04-08]. Dostupné z: https://www.doctrine-project.org

[6] Composer [online]. [cit. 2019-04-08]. Dostupné z: https://getcomposer.org

[7] Latte, nejbezpečnější a opravdu intuitivní šablony pro PHP [online]. [cit. 2019-04-08]. Dostupné z: https://latte.nette.org/cs/

[8] Twig, The flexible, fast, and secure template engine for PHP [online]. [cit. 2019-04-08]. Dostupné z: https://twig.symfony.com

[9] Symfony, high performance PHP framework for Web Development [online]. [cit. 2019-04-08]. Dostupné z: https://symfony.com

[10] Mark Otto, Jacob Thornton, and Bootstrap contributors: Bootstrap [online]. [cit. 2019-04-08]. Dostupné z: https://getbootstrap.com

[11] jQuery Foundation – jquery.org: jQuery [online]. [cit. 2019-04-08]. Dostupné z: http://jquery.com

[12] Apache, HTTP server project [online]. 1995 [cit. 2019-04-18]. Dostupné z: https://httpd.apache.org

[13] Apache Friends. XAMPP. Version 7.2 [software]. 2018. [cit. 2019-04-18]. Dostupné z: https://www.apachefriends.org. Multiplatformní.

[14] ČVUT Fakulta Informačních Technologií [online]. Architektonické vzory. 2018. [cit. 2019-05-05]. https://moodle.fit.cvut.cz/pluginfile.php/88287/mod_resource/content/0/06.prednaska.pdf.

[15] MAMP. Version 5.3 [software]. 2018. [cit. 2019-04-18]. Dostupné z: https://www.mamp.info/en/.

[16] The 8 Best Journal Apps for 2019 [online]. Lifewire. [cit. 2019-04-24]. Dostupné z: https://www.lifewire.com/best-journal-apps-4175848

[17] Journey Diary & Journal App [online]. Two App Studio Pte. Ltd., Singapore. [cit. 2019-04-24]. Dostupné z: https://2appstudio.com/journey/

[18] Timo Partl [online]. Timo Partl. [cit. 2019-04-24]. Dostupné z: https://timopartl.com

[19] Write In Private: Free Online Diary And Personal Journal | Penzu [online]. Penzu Inc. [cit. 2019-04-24]. Dostupné z: https://penzu.com

[20] DIARY and JOURNAL – Private writing with FREE APP! [online]. WriteDiary.com. [cit. 2019-04-24]. Dostupné z: https://www.writediary.com

[21] Day One – The award-winning journal app for iPhone, iPad, and Mac. [online]. Bloom Built, Inc. [cit. 2019-04-24]. Dostupné z: https://dayoneapp.com

[22] Five Minute Journal App – Intelligent Change [online]. Intelligent Change Inc. [cit. 2019-04-24]. Dostupné z: https://www.intelligentchange.com/pages/five-minute-journal-app?utm_source=zapier.com&utm_medium=referral&utm_campaign=zapier

[23] Diaro – Diary, Journal, Notes [online]. PIXEL CRATER LTD. 2010 – 2018 [cit. 2019-04-24]. Dostupné z: https://diaroapp.com

[24] GitHub – Kdyby/Translation: Integration of Symfony/Translation into Nette Framework [software]. GitHub, Inc. 2019 [cit. 2019-04-09]. Dostupné z: https://github.com/kdyby/Translation

[25] Jak na multijazyčný web s Nette Framework a Kdyby\Translation | Honza Černý / Blog [online]. HonzaCerny.com. [cit. 2019-04-28]. Dostupné z: http://blog.honzacerny.com/post/5-jak-na-multijazycny-web-s-nette-framework-a-kdyby-translation

[26] Vícejazyčný web [online]. [cit. 2019-04-28]. Dostupné z: http://rudolfsvatek.cz/vicejazycny-web

[27] Tagify.io, lightweight, efficient Tags input component in Vanilla JS / React / Angular [online]. [cit. 2019-04-08]. Dostupné z: https://github.com/yairEO/tagify

[28] Symfony, A JavaScript event calendar. Customizable and open source. [online]. [cit. 2019-04-08]. Dostupné z: https://fullcalendar.io

[29] Google Maps – Visible Markers In Bounds – JSFiddle [online]. [cit. 2019-04-11]. Dostupné z: https://jsfiddle.net/glafarge/mbuLw/

Příloha A

Seznam použitých zkratek

API Application Programming Interface

BI-SP1 Softwarový týmový projekt 1

BI-SP2 Softwarový týmový projekt 2

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheets

DAO Data Access Object

FP Funkční požadavek

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

JPEG Joint Photographic Experts Group

JSON JavaScript Object Notation

MVC Model View Controller

MVP Model View Presenter

MAMP Macintosh, Apache, MySQL, PHP

PDF Portable Document Format

PHP Hypertext Preprocessor

PNG Portable Network Graphics

REST Representational State Transfer

SQL Structured Query Language

UC Use Case

UML Unified Modeling Language

WYSIWYG What You See Is What You Get

XAMPP Cross-platform, Apache, MariaDB, PHP, Perl

XSS Cross-site scripting

Příloha B

Obsah přiloženého CD

- `readme.txt` ... stručný popis obsahu CD
- `release` ... adresář s výslednou verzí aplikce
  - `Journal-web-v1.0` ... soubory pro nasazení
  - `installGuide.pdf` ... příručka s návodem pro nasazení
  - `sql` ... složka s SQL skripty
- `src` ... zdrojové kódy implementace
  - `app` ... složka s jádrem aplikace
    - `config` ... složka s konfigurací aplikace
    - `exceptions` ... složka s vlastními výjimkami
    - `FrontModule` ... složka s webovou částí aplikace
    - `lang` ... složka s překlady pro aplikaci
    - `presenters` ... složka s BasePresenterem
    - `RestModule` ... složka s RESTovým rozhraním
    - `router` ... složka s mapováním aplikace
    - `templates` ... složka obsahující hlavní latte šablonu
    - `tests` ... složka s testy
  - `custom_vendor` ... složka s externími komponenty
  - `files` ... složka pro ukládání příloh
  - `install` ... skript pro nainstalování na serveru
  - `log` ... složka pro zaznamenávání chyb
  - `sql` ... složka s SQL skripty
  - `temp` ... složka s dočasnými soubory
  - `vendor` ... složka se závislostmi
  - `www` ... jediná přístupná složka na webu
    - `index.php` ... vstupní soubor aplikace
- `thesis` ... zdrojová forma práce ve formátu LaTeX
- `BP_Herman_Matyas_2019.pdf` ... text práce ve formátu PDF
- `příručky` ... příručky aplikace